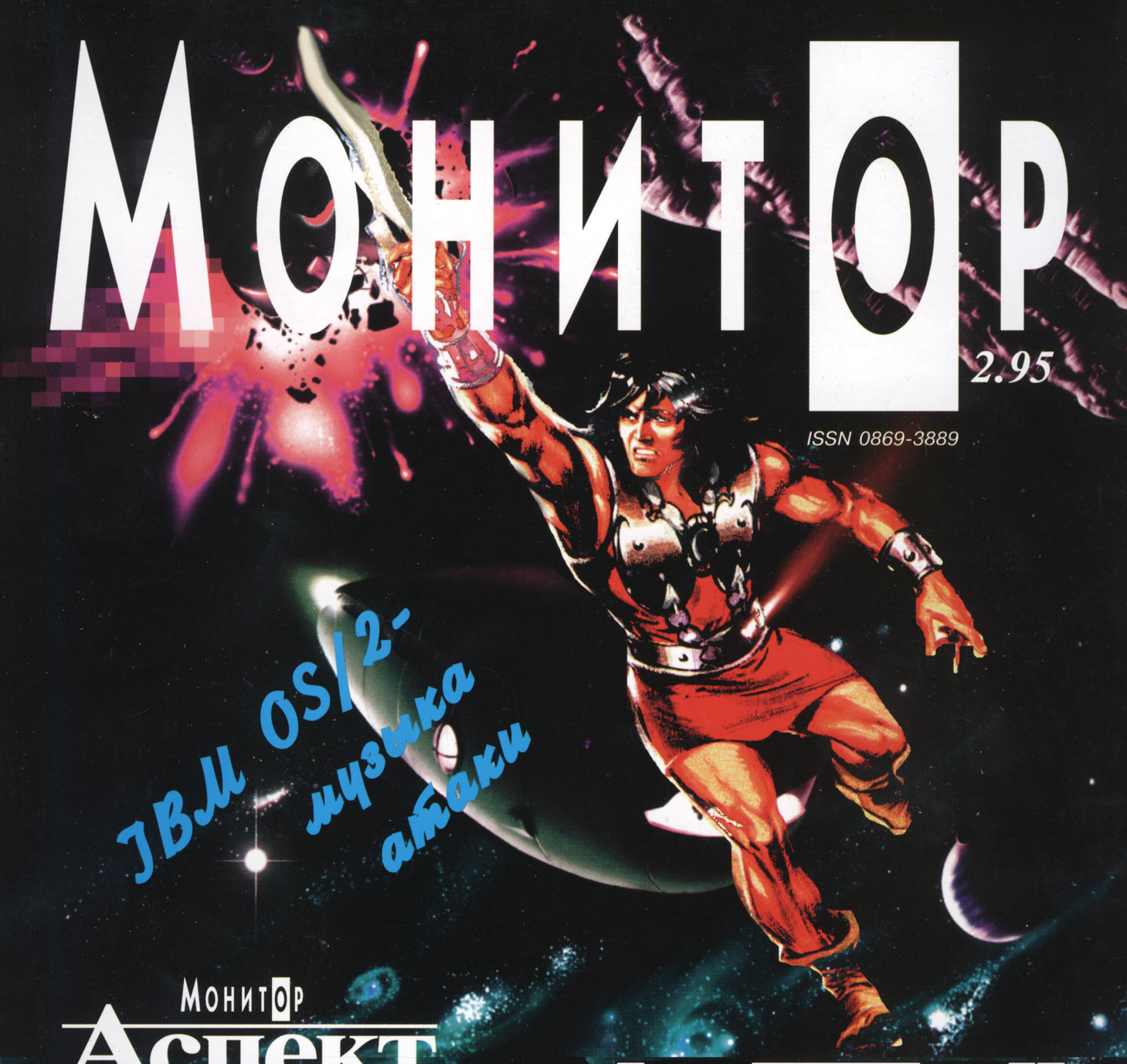
МОНИТОР

2.95

ISSN 0869-3889



IBM OS/2 - музыка атаки

МОНИТОР
Аспект

Программы

MathCAD - пять с плюсом

АППАРАТУРА

Система команд PowerPC 601

Pentium OverDrive

ТЕХНОЛОГИИ

Полиморфизм в среде xBase

Псевдомногозадачность в Turbo Vision

Автокад - это путь к прогрессу науки и техники.

Автокад - это ключ к успеху и процветанию.

Автокад - это не только универсальное средство компьютерного дизайна,

но и стимулятор интеллекта, так как именно Автокад заставляет Вас генерировать новые идеи и проекты.

Овладение Автокадом может стать для Вас увлекательным интеллектуальным приключением, если..... если у Вас есть Автокад!

Autodesk®

Телефон: 261 63 63
Факс: 267 99 27

Уважаемый читатель!

Представляя этот номер «Монитора», я хочу обратить Ваше внимание на статью «PC и Макинтош в графических операционных средах», подготовленную Независимой Тестовой Лабораторией. В статье приведены результаты первых для НТЛ испытаний компьютеров. Число моделей компьютеров, участвовавших в испытаниях, было сознательно ограничено — четыре PC и два Макинтоша — поскольку с каждым из компьютеров–участников была проведена большая работа и НТЛ стремилась получить максимально качественные, достоверные и подробные результаты. Мне кажется, что эти результаты должны оказаться интересными для Вас по двум причинам.

Во–первых, насколько мне известно, это первое в нашей стране тестирование компьютеров, проведенное сразу в четырех графических операционных средах: Windows 3.1, OS/2 и X–Windows (Linux) для PC и System 7 для Макинтоша. В статье приведены подробное описание методики тестирования, набора тестов, анализ аппаратной конфигурации компьютеров и ее влияния на производительность различных подсистем и, как следствие, на полученные результаты, и, конечно, результаты испытаний. Все это демонстрирует вдумчивому читателю подход к решению проблемы выбора компьютера, который должен эффективно работать в становящихся стандартом графических средах. В данном случае метод, как мне кажется, важнее конкретных результатов, поскольку он позволяет Вам самостоятельно проделать то же с теми моделями компьютеров, которые составляют Ваш индивидуальный спектр выбора.

Во–вторых, насколько мне известно, это первое проведенное в нашей стране объективное и независимое сравнение компьютеров PC и Макинтош на достаточно представительном наборе тестов. Ни в коем случае нельзя делать на его основе каких–либо глобальных выводов, сопоставляющих вообще PC и Макинтош, или процессоры Intel и PowerPC; полученные результаты относятся только к участвовавшим в испытаниях моделям, однако, на мой взгляд, они весьма интересны и дают серьезную пищу для размышлений.

Графические операционные среды неуклонно вытесняют DOS, и мы, естественно, уделяем им все больше внимания и стремимся помочь нашим читателям в их освоении: в этом номере вы найдете и рубрику «OS/2», и рубрику «Unix», и рубрику «Windows».

Вниманию работающих или намеревающихся работать на Макинтоше в этом номере предлагается первая статья из учебного цикла статей Михаила Трофимова, посвященного программированию на Макинтоше.

Мы продолжаем поддерживать отечественных разработчиков и публиковать статьи, рассказывающие о конкурентоспособных отечественных программах. С третьего номера мы начинаем публикацию цикла статей о пакете SuperVision, по ряду возможностей и быстродействию превосходящем широко известную систему TurboVision фирмы «Borland».

Для лучшего понимания того, какие темы представляют наибольший интерес для наших читателей, мы составили анкету (стр. 4). Мы приглашаем всех ответить на вопросы анкеты и прислать ее по адресу: 117419 Москва, а/я 765. Круг Ваших интересов и пожелания будут учтены при формировании последующих номеров «Монитора». Кроме того, среди участников анкетирования будет проведена лотерея, и победителей ждут призы!

Д.Э. Бройтман, главный редактор

Реклама в номере

Apple	45	Trans–Ameritech	65
ARUS	5, 105	Vimcom	3–я обл.
Autodesk	2–я обл.	YAM Int.	108
Boston PC	82	Интралекс	25
CROC, Inc.	4–я обл.	СП.АРМ	25
Crocus Int.	38–39	СП «Вариант»	78
DEC	119	Монолог	71
ELVIS +	51	Московский центр лазерной технологии	104
Novex	50	НАТЕКС	56, 104
NEXUS	17		
SSI	59		

Ответственность за информацию, приведенную в рекламных материалах, несет рекламодатель.

Наш компьютерный ларек вы найдете на стр. 122–123.

Редколлегия:
Д.Э. Бройтман
(главный редактор)
(brtmn@mntr.msk.su)
В.В. Водолазский
(vodolaz@mnyt.msk.su)
В.С. Головин
Н.Н. Косарев
А.В. Лепехин
(отдел новостей)
А.Р. Мурадян
(зав. тестовой лабораторией)
О.А. Пригоникер
Е.В. Равич
Л.И. Турок
(turok@mntr.msk.su)

Отдел рекламы:
Н.А. Никитина
(заведующая отделом)
В.Ю.Томская–Ефремова

Графическое бюро:
С.В. Макельский
(главный художник)
Д.В. Герасимова
Ю.А. Исаева
А.В. Семенников
(компьютерная верстка)

Компьютерный «ларек»:
А.В. Лепехин (lav@mntr.msk.su)

Отдел распространения:
М.Е. Желаннова
(заведующая отделом)
Р.А. Болдырев
А.А. Мишустин
М.А. Сычев

Дирекция:
А.В. Орехов
(издательский директор)
А.Д. Гассельблат
(коммерческий директор)
Л.Я. Кус
(зам. директора)

В оформлении номера использованы рисунки
А.С. Замулы

Для писем:	**117419 Москва, а/я 765**
E–mail:	**post@mntr.msk.su**
Редакция:	**(095) 238–99–33, (095) 238–31–89**
Факс:	**(095) 238–99–11**
Отдел рекламы:	**(095) 903–46–26**

Учредитель:
АО "Монитор"

Журнал зарегистрирован Комитетом Российской Федерации по печати 20.07.94. Регистрационный номер 012752

Подписано в печать с оригинал–макета 03.03.95
Формат 60x84/8.
Отпечатано в типографии «Oy Scanweb Ab» (Финляндия)

© «Монитор», 1995

Процессоры	Д. Бройтман.	Микропроцессор PowerPC 601 Часть 2. Система команд	6
		Процессор Pentium OverDrive	8
Операционные системы	А. Козленков.	IBM OS/2 Warp — музыка атаки	18
Windows	В. Плотников.	Использование возможностей защищенного режима работы процессора в Windows - приложениях	26
UNIX	Н. Кожекин.	Начала программирования в XWindow	40
Языки программирования	А. Тихонов.	Система программирования Stony Brook Pascal+	46
Базы данных	А. Дубина, С. Сынгур.	Процедурные файлы — платформа для реализации полиморфизма в среде xBase	52
MS BASIC	А. Колесов.	Нам пишут, мы отвечаем	57
Turbo Vision	С. Миллер.	Организация псевдомногозадачности в Turbo Vision	60
Программирование на Макинтоше	М. Трофимов.	Жизнь на Макинтоше: по страницам новой книги Часть 1	72

Аппаратура

Настольные компьютеры

А. Мурадян.
PC и Макинтош в графических операционных средах — 84

Храните ваши байтики... — 92

Мультимедиа

В. Дьяконов.
Базовые средства мультимедиа в Microsoft Windows — 96

Печатающие устройства

В. Водолазский.
Струйный принтер — экзотика или рабочий инструмент? — 102

Сканеры

Д. Мокин.
Сканеры фирмы UMAX — 106

PC

Персональные компьютеры — 84, 96, 109

Макинтоши

84, 124

Программы

Математические системы

В. Дьяконов.
MathCAD — пять с плюсом! — 109

Технологии

Сети

Р. Убайдуллаев.
Сети нового поколения: технология FDDI — 116

Это сладкое слово «надежность» — 120

Ad hominem

Образование

А. Горчаков.
Компьютерные решения в гуманитарном образовании — 124

Новости

127

Анкета «Монитора»

С целью лучшего учета интересов и направлений деятельности читателей «Монитора» при формировании содержания журнала редакция «Монитора» проводит анкетирование. Мы приглашаем всех наших читателей ответить на приведенные далее вопросы и выслать заполненную анкету по адресу 117419 Москва, а/я 765. Ваши ответы помогут нам при работе над последующими номерами «Монитора». Если отведенного в анкете места мало для выражения Ваших пожеланий, мы будем рады получить более развернутые ответы и письма.

Среди участников анкетирования будет проведена лотерея и разыграны призы (годовая подписка на «Монитор», компьютерные аксессуары, игры).

1. Ваше место жительства (город)?

2. Ваш возраст?

3. К какой профессиональной группе Вы относите себя?
   - ☐ школьник
   - ☐ студент
   - ☐ инженер
   - ☐ менеджер
   - ☐ руководитель
   - ☐ другое (пожалуйста, укажите)

4. Как Вы оцениваете свою компьютерную квалификацию?
   - ☐ начинающий пользователь
   - ☐ квалифицированный пользователь
   - ☐ начинающий программист
   - ☐ программист-любитель
   - ☐ квалифицированный программист

5. Какое из следующих утверждений справедливо для Вас (можно выбирать несколько)?
   - ☐ Вы не программируете, а только используете компьютер на работе, дома или в учебе
   - ☐ программирование — Ваше хобби
   - ☐ Вы используете программирование как средство для решения задач, возникающих в Вашей основной деятельности
   - ☐ программирование — Ваше основное занятие
   - ☐ Вы используете компьютер, в том числе для отдыха и развлечений

6. С каким компьютером Вы обычно работаете?
   а) тип компьютера
   - ☐ PC
   - ☐ Макинтош
   - ☐ другой

   б) процессор

   в) объем оперативной памяти

   г) CD-ROM (есть/нет)

7. С какой операционной системой Вы работаете?

8. Собираетесь ли Вы сменить операционную систему (если да, то на какую)?

9. Какие языки программирования Вы предпочитаете (укажите языки и компиляторы)?

10. Какие СУБД Вы используете?

11. Представляют ли, по Вашему мнению, интерес публикации по работе в среде DOS?

12. Какие рубрики в «Мониторе» Вы читаете?
   - ☐ Технологии программирования
   - ☐ Операционные системы
   - ☐ Процессоры
   - ☐ Базы данных
   - ☐ Языки программирования
   - ☐ MS Basic
   - ☐ DOS под микроскопом
   - ☐ Мир графики
   - ☐ Защита
   - ☐ Коммуникации
   - ☐ Windows
   - ☐ Unix
   - ☐ OS/2
   - ☐ Turbo Vision
   - ☐ Лицей
   - ☐ Письма

13. С каким из следующих утверждений Вы согласны?
   - ☐ Листинги программ печатать в «Мониторе» не следует, надо ограничиться небольшими фрагментами листингов внутри статьи.
   - ☐ Листинги надо печатать, но в меньшем объеме, чем сейчас.
   - ☐ Листинги программ должны занимать столько же места, что и сейчас, или даже больше.
   - ☐ Все листинги следует помещать только в «Монитор-Диск» с целью увеличения количества статей. При этом Вы готовы нести дополнительные расходы, связанные с получением журнала в комплекте с дискетой..
   - ☐ Другое

14. Какие из перечисленных далее алгоритмов интересны для Вас?
   - ☐ Алгоритмы сжатия информации
   - ☐ Алгоритмы обработки изображений
   - ☐ Алгоритмы защиты информации
   - ☐ Алгоритмы шифрования
   - ☐ Алгоритмы распознавания образов
   - ☐ Другие
   - ☐ Алгоритмы мне неинтересны

15. Читаете ли Вы журнал в журнале «Монитор-Аспект» (да или нет)?

16. Какие рубрики «Монитор-Аспекта» интересны для Вас?
   - ☐ Настольные компьютеры
   - ☐ Рабочие станции
   - ☐ Специализированные компьютеры
   - ☐ Портативные компьютеры
   - ☐ Текстовые редакторы
   - ☐ Электронные таблицы
   - ☐ Интегрированные среды
   - ☐ Математические системы
   - ☐ Мир графики
   - ☐ Издательские системы
   - ☐ Тестовая лаборатория
   - ☐ Деловому человеку
   - ☐ Мультимедиа
   - ☐ САПР
   - ☐ Сети
   - ☐ Образование
   - ☐ Здоровье
   - ☐ Игры
   - ☐ Новости

17. Результаты каких сравнительных тестирований программного обеспечения и оборудования представляют для Вас максимальный интерес?

18. Представляют ли для Вас интерес рекламные материалы? В какой степени?
   - ☐ Для связи с рекламодателями
   - ☐ В качества источника информации (включение в базу данных и т. п.)
   - ☐ Другое
   - ☐ В чисто ознакомительных целях
   - ☐ Не интересуют

19. Хотели бы Вы получать от фирм-рекламодателей бесплатные рекламные материалы (да/нет)?

20. Какие материалы информационно-рекламного характера Вы хотели бы видеть на страницах «Монитора»?

21. Какое из следующих утверждений, по Вашему мнению, справедливо?
   - ☐ Изложение в «Мониторе» слишком серьезно и математизировано
   - ☐ Изложение в « Мониторе» слишком популярно
   - ☐ Уровень изложения меня устраивает

22. Какие важные, на Ваш взгляд, темы, не находят отражения в «Мониторе»?

23. Каким темам, по Вашему мнению, «Монитор» уделяет неоправданно много внимания?

24. Что бы Вы посоветовали редакции «Монитора»?

25. Ваши любимые авторы статей в «Мониторе»?

26. Не хотите ли Вы предложить свои статьи для публикации в «Мониторе» (если да, то какие)?

27. Какие еще компьютерные журналы Вы читаете?
   - ☐ «Компьютер Пресс»
   - ☐ «PC Magazine/Russian Edition»
   - ☐ «Dr. Dobb's Journal»
   - ☐ «Мир ПК»
   - ☐ «Byte»
   - ☐ «PC Magazine»
   - ☐ Другие

28. Координаты для связи с Вами (в случае выигрыша приза)

ARUS MOSCOW

HEWLETT PACKARD
Authorized Wholesaler

НАДЕЖНОСТЬ, ПРОВЕРЕННАЯ ВРЕМЕНЕМ

Компьютеры, серверы, сканеры, лазерные и струйные принтеры, сетевое оборудование фирмы *HEWLETT-PACKARD*

Наши региональные дилеры:

Москва:	**"ARUS-Тех. Центр"**	(095) 316-7627	(095) 316-8328
	"K&M"	(095) 323-9366	(095) 323-9172
	"E+E"	(095) 916-1449	(095) 917-3379
	"EuroNet"	(095) 281-0178	(095) 281-4141
	"DiViSy"	(095) 460-4723	(095) 460-0212
	"Синэус"	(095) 318-0722	(095) 318-6324
	"Delta Group"	(095) 241-6968	(095) 241-6968
	"Смарт"	(095) 556-4379	(095) 556-4262
Ангарск:	**"Форт-Диалог"**	(39518) 5-7653	(39518) 6-3216
Казань:	**"Абак"**	(8432) 322-023	(8432) 323-061
Наб. Челны:	**"Форт-Диалог"**	(8439) 56-1467	(8439) 56-8129
Новосибирск:	**"Велл"**	(3832) 328-857	(3832) 391-527
Одесса:	**"ARUS-Odessa"**	(0482) 668-090	(0482) 657-055
Самара:	**"Резольвента"**	(8462) 222-996	(8462) 222-996
Тольятти:	**"Пролог +"**	(8469) 346-730	(8469) 341-183
Уфа:	**"Арти-Системс"**	(3472) 35-6280	(3472) 33-0535
Юж.-Сахал.:	**"Сахин"**	(42422) 31-478	(42422) 31-397

Продолжаем формирование дилерской сети

316-7627, 316-8328, 110-5830, 230-6808. Fax: 119-6841

Гарантия HEWLETT-PACKARD до 3-х лет

Микропроцессор PowerPC 601

Д. Бройтман

Часть 2. Система команд

Статья содержит описание системы команд микропроцессора PowerPC 601.

1. Архитектура — основные элементы

Архитектура микропроцессора PowerPC 601 (далее просто 601) подробно описана в первой части настоящей статьи [1]. Здесь мы повторим описание только тех составляющих архитектуры 601, которые упоминаются при дальнейшем изложении.

601 имеет два режима работы: *режим пользователя и режим супервизора*. В режиме супервизора осуществляется управление состоянием процессора, механизмом трансляции адресов и регистрами супервизора.

Регистры

32 регистра общего назначения GPR[0..31].

32 регистра с плавающей запятой FPR[0..31].

Регистр условия CR — 8 полей по 4 бита. Содержание полей определяется результатами выполнения команд.

Регистр состояния и управления для операций с плавающей запятой FPSCR.

Регистр состояния машины MSR. Доступен в режиме супервизора. Сохраняется при обработке исключительных ситуаций.

16 регистров сегмента SR[0..15]. Доступны в режиме супервизора для управления памятью.

Регистр связи LR. Используется в командах перехода.

Регистр-счетчик CTR. Используется в командах перехода.

Регистр целочисленного исключения XER. Используется в исключительных ситуациях (перенос, переполнение и тому подобные).

Регистры часов реального времени RTCU и RTCL. Чтение — в режиме пользователя, запись — только в режиме супервизора.

Регистр расширения MQ. Используется в командах умножения, деления, сдвига и вращения, унаследованных от архитектуры POWER. В архитектуру PowerPC не входит.

Большое число регистров специального назначения уровня супервизора.

Размерность всех регистров, за исключением FPR, — 32 бита. Размерность FPR — 64 бита.

Организация памяти

В 601 реализован 32-битовый логический адрес. 4 гигабайта (2^{32} байта) логического адресного пространства доступны и в режиме супервизора, и в режимепользователя.

Логическое адресное пространство делится на страницы, сегменты и блоки.

Страница — 4 килобайта (2^{12} байта).

Сегмент — 256 мегабайтов (2^{28} байта).

4 Гб / 256 Мб = 16 сегментов.

Блоки могут иметь размер от 128 кб (2^{17} байта) до 8 Мб (2^{23} байта).

Для каждого блока или страницы операционная система создает дескриптор, в котором указаны местоположение блока (страницы) в физической памяти и сведения о правилах доступа к блоку (странице). Дескрипторы хранятся в таблицах в памяти или в специальных буферах процессора.

При трансляции адреса 32-битовый логический адрес преобразуется в 52-битовый виртуальный, а тот в свою очередь — в 32-битовый физический адрес. Трансляции подвергаются старшие 20 битов (адрес страницы); младшие 12 битов (адрес внутри страницы) не меняются. По физическому адресу операнд или команда отыскиваются в кэше, в оперативной памяти или где-нибудь еще.

Устройство управления памятью содержит внутренние флаги «выборка команд/выборка данных» и «супервизор/пользователь».

Режимы защиты памяти приведены в таблице 1.

Защита накладывается на целые страницы или блоки.

2. Общие сведения о командах и операндах

Размерность целых операндов может составлять 8, 16 и 32 бита (байт, полуслово и слово); размерность ве-

Таблица 1

Режимы защиты памяти PowerPC 601

Режим \ возможности	Чтение — пользователь	Запись — пользователь	Чтение — супервизор	Запись — супервизор
Только супервизор	-	-	+	+
Запись — только супервизор	+	-	+	+
Пользователь/ супервизор	+	+	+	+
Только чтение	+	-	+	-

щественных операндов — 32 и 64 бита (слово и двойное слово).

Все команды имеют размер 32 бита и выровнены по границе слова. Биты 0-5 всегда содержат первичный код операции. Ряд команд включает также поле вторичного кода операции. Биты 6-31 делятся на поля, различные для разных групп команд.

Подавляющее большинство команд в 601 выполняется за один цикл.

Команды делятся на следующие *группы*:

- целые команды (арифметические и логические);
- команды с плавающей точкой (арифметические и смена состояния FPSCR);
- команды обмена с памятью;
- команды, изменяющие порядок выполнения программы;
- команды управления процессором;
- команды управления памятью;
- команды внешнего управления.

Часть команд 601 унаследована от более ранней архитектуры POWER и поддерживается в целях совместимости. В последующих версиях микропроцессора PowerPC эти команды будут исключены. В настоящей статье описание таких команд не приводится.

Цель настоящей статьи — краткое описание системы команд 601; полное описание содержится в руководстве [2] (будьте внимательны: там есть опечатки!).

Используемые при описании команд условные обозначения приведены во врезке на стр. 16.

3. Целые команды

Целые команды делятся на четыре группы: арифметические, сравнения, сдвига и вращения, логические.

В качестве операндов-источников они используют GPR[0..31] и непосредственный операнд. Результаты помещаются в GPR[0..31], XER и поля CR.

Операнды-источники обрабатываются как целые со знаком, за исключением тех случаев, когда в команде явно обозначена операция без знака.

К базовым именам команд могут добавляться суффиксы «.» и «o». Суффикс «.» означает, что в соответствии с результатом команды будет обновлено поле CR0 (биты 0-3) регистра условия CR. Содержимое этого поля соответствует результату алгебраического сравнения результата команды с нулем. Суффикс «o» означает, что биты SO и OV регистра XER будут установлены в случае переполнения результата.

Целые арифметические команды addic, addic., subfic, addc, subfc, adde, subfe, addme, subfme, addze, subfze устанавливают бит CA в регистре XER в случае переноса из бита 0.

3.1. Целые арифметические команды

1) Add Immediate
   addi rD,rA,SIMM
   rD ← (rA|0) + SIMM
2) Add Immediate Shifted
   addis rD,rA,SIMM
   rD ← (rA|0) + (SIMM || x'0000')
3) Add
   add[o][.] rD,rA,rB
   rD ← (rA) + (rB)
4) Subtract from
   subf[o][.] rD,rA,rB
   rD ← not(rA) + (rB) + 1
5) Add Immediate Carrying
   addic[.] rD,rA,SIMM
   rD ← (rA) + SIMM
6) Subtract from Immediate Carrying
   subfic rD,rA,SIMM
   rD ← not(rA) + SIMM + 1
7) Add Carrying
   addc[o][.] rD,rA,rB
   rD ← (rA) + (rB)
8) Subtract from Carrying
   subfc[o][.] rD,rA,rB
   rD ← not(rA) + (rB) + 1
9) Add Extended
   adde[o][.] rD,rA,rB
   rD ← (rA) + (rB) + XER[CA]
10) Subtract from Extended
   subfe[o][.] rD,rA,rB
   rD ← not(rA) + (rB) + XER[CA]
11) Add to Minus One Extended
   addme[o][.] rD,rA
   rD ← (rA) + XER[CA] + x'FFFFFFFF'
12) Subtract from Minus One Extended
   subfme[o][.] rD,rA
   rD ← not(rA) + XER[CA] + x'FFFFFFFF'
13) Add to Zero Extended
   addze[o][.] rD,rA
   rD ← (rA) + XER[CA]
14) Subtract from Zero Extended
   subfze[o][.] rD,rA
   rD ← not(rA) + XER[CA]
15) Negate
   neg[o][.] rD,rA
   rD ← not(rA) + 1
16) Multiply Low Immediate
   mulli rD,rA,SIMM
   rD ← младшие 32 бита 48-битового (rA) * SIMM
   Старшие 16 битов теряются. Младшие 32 бита не зависят от знаков операндов, однако XER[OV] устанавливается, рассматривая результат как целое со знаком.
   При использовании mulli вместе с mulhw может быть получено полное 64-битовое произведение.
17) Multiply Low
   mullw[o][.] rD,rA,rB
   rD ← младшие 32 бита 64-битового (rA) * (rB)
   Старшие 32 бита теряются. Младшие 32 бита не зависят от знаков операндов, однако XER[OV] устанавливается, рассматривая результат как целое со знаком.
   При использовании mullw вместе с mulhw может быть получено полное 64-битовое произведение.
18) Multiply High Word
   mulhw[.] rD,rA,rB
   rA и rB рассматриваются как целые со знаком.
   rD ← старшие 32 бита 64-битового (rA) * (rB)
19) Multiply High Word Unsigned
   mulhwu[.] rD,rA,rB
   rA и rB рассматриваются как целые без знака.
   rD ← старшие 32 бита 64-битового (rA) * (rB)
20) Divide Word
   divw[o][.] rD,rA,rB
   rA и rB рассматриваются как целые со знаком.
   rD ← целое частное (rA) / (rB)
   Нельзя делить на 0 и делить x'80000000' на -1.
   Вычисление остатка:
   divw rD,rA,rB mullw rD,rD,rB subf rD,rD,rA
21) Divide Word Unsigned
   divw[o][.] rD,rA,rB
   rA и rB рассматриваются как целые без знака.
   rD ← целое частное (rA) / (rB)

Нельзя делить на 0.
Вычисление остатка:
divwu rD,rA,rB mullw rD,rD,rB subf rD,rD,rA

3.2. Целые команды сравнения

Для реализации заложенной в архитектуре PowerPC возможности работы с 64-битовыми операндами в целых командах сравнения введено поле L (0 — 32-битовые операнды, 1 — 64-битовые). В 601 поле L игнорируется, и поэтому для целых команд сравнения существует сокращенная мнемоника, приводимая после основной.

crfD задает одно из 8 полей регистра CR, в которое записывается результат сравнения. По умолчанию (если crfD опущено) это CR0.

1) Compare Immediate
   cmpi crfD,L,rA,SIMM cmpwi crfD,rA,SIMM
   crfD ← результат алгебраического сравнения (rA) и SIMM
2) Compare
   cmp crfD,L,rA,rB cmpw crfD,rA,rB
   crfD ← результат алгебраического сравнения (rA) и (rB)
3) Compare Logical Immediate
   cmpli crfD,L,rA,UIMM cmplwi crfD,rA,UIMM
   crfD ← результат сравнения (rA) и (x'0000' || UIMM) как целых без знака.
4) Compare Logical
   cmpl crfD,L,rA,rB cmplw crfD,rA,rB
   crfD ← результат сравнения (rA) и (rB) как целых без знака.

3.3. Целые логические команды

1) AND Immediate
   andi. rA,rS,UIMM
   rA ← (rS) & (x'0000' || UIMM)
2) AND Immediate Shifted
   andis. rA,rS,UIMM
   rA ← (rS) & (UIMM || x'0000')
3) OR Immediate
   ori rA,rS,UIMM
   rA ← (rS) | (x'0000' || UIMM)
   В качестве пустой операции рекомендуется ori 0,0,0
4) OR Immediate Shifted
   oris rA,rS,UIMM
   rA ← (rS) | (UIMM || x'0000')
5) XOR Immediate
   xori rA,rS,UIMM
   rA ← (rS) xor (x'0000' || UIMM)
6) XOR Immediate Shifted
   xoris rA,rS,UIMM
   rA ← (rS) xor (UIMM || x'0000')
7) AND
   and[.] rA,rS,rB
   rA ← (rS) & (rB)
8) OR
   or[.] rA,rS,rB
   rA ← (rS) | (rB)
9) XOR
   xor[.] rA,rS,rB
   rA ← (rS) xor (rB)
10) NAND
   nand[.] rA,rS,rB
   rA ← not [(rS) & (rB)]
11) NOR
   nor[.] rA,rS,rB
   rA ← not [(rS) | (rB)]
12) Equivalent
   eqv[.] rA,rS,rB
   rA ← not [(rS) xor (rB)]
13) AND with Complement
   andc[.] rA,rS,rB
   rA ← (rS) & not(rB)
14) OR with Complement
   orc[.] rA,rS,rB
   rA ← (rS) | not(rB)
15) Extend Sign Byte
   extsb[.] rA,rS
   rA[24..31] ← rS[24..31] rA[0],rA[1],..,rA[23]← rS[24]
16) Extend Sign Half Word
   extsh[.] rA,rS
   rA[16..31] ← rS[16..31] rA[0],rA[1],..,rA[15]← rS[16]
17) Count Leading Zeros Word
   cntlzw[.] rA,rS
   rA ← число последовательных нулей, начиная с бита 0, в rS

3.4. Целые команды вращения и сдвига

Операнды MB и ME задают маску: 32-битовый операнд, содержащий 1 в битах MB..ME и 0 в остальных битах.

Процессор Pentium OverDrive

23 января 1995 года «Intel Corp.» объявила о начале поставок нового процессора семейства OverDrive — процессора Pentium OverDrive.

Семейство процессоров OverDrive поддерживает технологию модернизации систем на базе архитектуры Intel. Наилучшим решением для пользователя Intel486 компьютера, стремящегося увеличить производительность своей работы, фирма «Intel» считает покупку нового компьютера с процессором Pentium, что обеспечивает сочетание самого производительного процессора Intel и новейших разработок в области шин и памяти. Для тех же, кто не имеет возможности приобрести новую систему на базе Pentium, «Intel» рекомендует замену центрального процессора на Pentium OverDrive.

Pentium OverDrive изготовлен по новейшей технологии Intel (3,3 В, 0,6 мкм). Он сохраняет основные особенности Pentium — суперскалярность, предсказание переходов, быстрое устройство для команд с плавающей запятой. Его отличия от Pentium — больший размер кэша первого уровня (32 килобайта), 32-битовый интерфейс шины, оптимизированный размер строки кэша и отношение тактовых частот ядра и шины, равное 5:2. Кроме того, процессор заключен в новый корпус с интегрированными стабилизатором напряжения и вентилятором-радиатором.

«Intel» специально подчеркивает, что обнаруженная недавно ошибка при выполнении процессором Pentium команды деления с плавающей запятой не распространяется на Pentium OverDrive, который выполняет эту команду правильно.

В настоящее время доступна первая модель Pentium OverDrive, предназначенная для замены процессоров с тактовой частотой шины 25 МГц — 50 МГц Intel SX2, IntelDX2 и 25 МГц Intel486. Ее внутренняя тактовая частота равна 63 МГц (25 * 5/2).

Ближе к середине 1995 года начнутся поставки второй модели — для замены процессоров с тактовой частотой шины 33 МГц (66 МГц IntelDX2 и 33 МГц Intel486). Ее внутренняя тактовая частота равна 83 МГц (33 * 5/2).

Pentium OverDrive поставляется в корпусе с 235 контактами, который устанавливается либо в гнездо Socket 3 (237 контактов), либо в Socket 2 (238 контактов). Большинство систем на основе процессоров Intel486 может быть модернизировано с помощью Pentium OverDrive, однако единственной гарантией возможности установки того или иного процессора OverDrive в конкретную систему является ее наличие в справочнике Intel Verified Systems Selection Guide, который ежемесячно переиздается компанией «Intel». Надпись на компьютере «Pentium OverDrive Ready» такой гарантии не дает, так как неудачное проектирование

1) Rotate Left Word Immediate then AND with Mask
   rlwinm[.] rA,rS,SH,MB,ME
   rA ← (rS вращается влево на SH битов) & маска
2) Rotate Left Word then AND with Mask
   rlwnm[.] rA,rS,rB,MB,ME
   rA ← (rS вращается влево на rB[27..31] битов) & маска
3) Rotate Left Word Immediate then Mask Insert
   rlwimi[.] rA,rS,SH,MB,ME
   rS вращается влево на SH битов, затем биты, задаваемые маской, помещаются на соответствующие позиции в rA, остальные биты rA не меняются.
4) Shift Left Word
   slw[.] rA,rS,rB
   rA ← (rS, сдвинутый влево на rB[27..31] битов)
   Освободившиеся биты заполняются нулями. Если rB[26] = 1, то весь регистр rA заполняется нулями.
5) Shift Right Word
   srw[.] rA,rS,rB
   rA ← (rS, сдвинутый вправо на rB[27..31] битов)
   Освободившиеся биты заполняются нулями. Если rB[26] = 1, то весь регистр rA заполняется нулями.
6) Shift Right Algebraic Word Immediate
   srawi[.] rA,rS,SH
   rA ← (rS, сдвинутый вправо на SH битов)
   Освободившиеся биты заполняются битом знака (rS[0]). Если (rS) < 0 и при сдвиге была потеряна хотя бы одна единица, то устанавливается XER[CA]. Иначе XER[CA] очищается.
7) Shift Right Algebraic Word
   sraw[.] rA,rS,rB
   rA ← (rS, сдвинутый вправо на rB[27..31] битов)
   Если rB[26] = 1, то rA полностью заполняется битом rS[0]. Если rB[26] = 0, то освободившиеся слева биты заполняются битом rS[0]. Если (rS) < 0 и при сдвиге была потеряна хотя бы одна единица, то устанавливается XER[CA]. Иначе XER[CA] очищается.

4. Команды с плавающей запятой

Команды с плавающей запятой делятся на пять групп: арифметические, умножения-сложения, округления и преобразования, сравнения, управления FPSCR.

В качестве операндов-источников они используют FPR[0..31]. Результаты помещаются в FPR[0..31], FPSCR и поля CR.

Команды с плавающей запятой реализованы в 601 в соответствии со стандартом IEEE-754. Подробное описание этого стандарта и деталей его реализации не входит в задачи настоящей статьи. Такое описание приведено в [2].

В архитектуре PowerPC предусмотрено два формата для чисел с плавающей запятой: с одинарной точностью (рис. 1) и с двойной точностью (рис. 2).

Операнды команд с двойной точностью могут иметь как одинарную, так и двойную точность.

Операнды команд с одинарной точностью могут иметь только одинарную точность.

Результат выполнения команды с двойной точностью — всегда число с двойной точностью.

Результат выполнения команды с одинарной точностью — всегда число с одинарной точностью.

При выполнении команд преобразование числа с одинарной точностью в число с двойной точностью может происходить неявно, а обратное преобразование всегда должно быть задано явно с помощью соответствующей команды.

К базовым именам команд может добавляться суффикс «.», который означает, что в соответствии с результатом команды будет обновлено поле CR0 (биты 0-3) регистра условия CR.

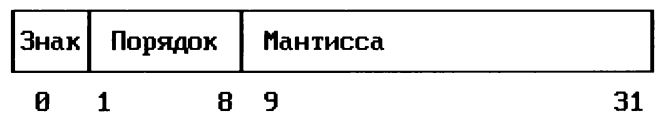


Рис. 1. Формат с одинарной точностью для чисел с плавающей запятой

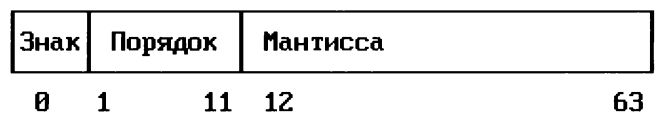


Рис. 2. Формат с двойной точностью для чисел с плавающей запятой

конкретной системы может привести к невозможности работы с процессором OverDrive. Например, у некоторых компьютеров BIOS, использующая циклы ожидания в виде последовательности команд, может неверно работать с новым процессором, результатом чего явится «зависание» компьютера.

Одновременно с началом поставок Pentium OverDrive «Intel» снизила рекомендуемые производителям цены на все остальные процессоры семейства OverDrive. В настоящее время это семейство включает следующие процессоры:

50 МГц IntelDX2 OverDrive по цене $149 (было $199);
66 МГц IntelDX2 OverDrive по цене $249 (было $299);
75 МГц IntelDX4 OverDrive по цене $299 (было $549);
100 МГц IntelDX4 OverDrive по цене $349 (было $649);
63 МГц Pentium OverDrive по цене $449.

В стадии разработки находятся процессоры OverDrive для замены 60 МГц, 66 МГц, 75 МГц, 90 МГц и 100 МГц процессоров Pentium.

При сравнении производительности 63 МГц Pentium OverDrive с 50 МГц IntelDX2 различные индексы производительности дают рост производительности примерно в 1,5 раза на целых операциях и более чем 2,5 раза на операциях с плавающей запятой.

Сравнение производительности пока еще официально не поставляемого 83 МГц Pentium OverDrive с 66 МГц IntelDX2 было проведено автором в московском представительстве «Intel». В качестве базы для сравнения был выбран компьютер на основе платформы Classic_R+ фирмы «Intel» с процессором 486DX2-66, 16 Мбайтами оперативной памяти, винчестером Maxtor MXT-7245AT емкостью 245 Мбайтов и видеоподсистемой на основе кристалла Cirrus Logic 5428 с видеопамятью 1 Мбайт. Использовались набор тестов журнала «Byte», набор тестов журнала «PC Magazine», тест производительности в среде Windows wintach и три специально подготовленных тестовых сценария для среды Windows (S1 — редактирование и фильтрация изображения в программе Adobe Photoshop с последующим импортом его в документ AmiPro, S2 — вычисления в электронной таблице Excel с последующим представлением результатов в виде диаграмм, S3 аналогичен S2, но реализован для Lotus 1-2-3). После прогона тестов процессор 486DX2-66 был заменен на 83 МГц Pentium OverDrive, и тесты были повторены.

Для того чтобы получить точку отсчета, соответствующую, на взгляд автора, средней технической оснащенности наших читателей, все тесты, за исключением последних трех, были проведены также для редакционного компьютера 386/387DX-40 с оперативной памятью 4 Мбайта, винчестером 120 Мбайтов и видеопамятью 512 килобайтов.

Результаты приведены в таблице 1. Как видно из таблицы, рост производительности для команд с плавающей точкой выше, чем для целых команд; рост производительности для программ, работающих в DOS, выше, чем для программ, работающих под Windows.

Д.Б.

Стр. 11

Содержимое этого поля соответствует результату алгебраического сравнения результата команды с нулем.

4.1. Арифметические команды с плавающей запятой

1) Floating-Point Add
   fadd[.] frD,frA,frB
   frD ← frA + frB
2) Floating-Point Add Single-Precision
   fadds[.] frD,frA,frB
   frD ← frA + frB
3) Floating-Point Subtract
   fsub[.] frD,frA,frB
   frD ← frA — frB
4) Floating-Point Subtract Single-Precision
   fsubs[.] frD,frA,frB
   frD ← frA — frB
5) Floating-Point Multiply
   fmul[.] frD,frA,frC
   frD ← frA * frC
6) Floating-Point Multiply Single-Precision
   fmuls[.] frD,frA,frC
   frD ← frA * frC
7) Floating-Point Divide
   fdiv[.] frD,frA,frB
   frD ← frA / frB
8) Floating-Point Divide Single-Precision
   fdivs[.] frD,frA,frB
   frD ← frA / frB

4.2. Команды умножения-сложения с плавающей запятой

Эти команды объединяют операции умножения и сложения (вычитания), при этом промежуточный результат не округляется. Под дробную часть промежуточного результата отводится 106 битов, и все 106 битов принимают участие во второй фазе команды.

1) Floating-Point Multiply-Add
   fmadd[.] frD,frA,frC,frB
   frD ← (frA * frC) + frB
2) Floating-Point Multiply-Add Single-Precision
   fmadds[.] frD,frA,frC,frB
   frD ← (frA * frC) + frB
3) Floating-Point Multiply-Subtract
   fmsub[.] frD,frA,frC,frB
   frD ← (frA * frC) — frB
4) Floating-Point Multiply-Subtract Single-Precision
   fmsubs[.] frD,frA,frC,frB
   frD ← (frA * frC) — frB
5) Floating-Point Negative Multiply-Add
   fnmadd[.] frD,frA,frC,frB
   frD ← — [(frA * frC) + frB]
6) Floating-Point Negative Multiply-Add Single-Precision
   fnmadds[.] frD,frA,frC,frB
   frD ← — [(frA *frC) + frB]
7) Floating-Point Negative Multiply-Subtract
   fnmsub[.] frD,frA,frC,frB
   frD ← — [(frA * frC) — frB]
8) Floating-Point Negative Multiply-Subtract Single-Precision
   fnmsubs[.] frD,frA,frC,frB
   frD ← — [(frA * frC) — frB]

4.3. Команды округления и преобразования с плавающей запятой

1) Floating-Point Round to Single-Precision
   frsp[.] frD,frB
   frD ← (frB), округленное до одинарной точности
   Режим округления задается битом FPSCR[RN].
2) Floating-Point Convert to Integer Word
   fctiw[.] frD,frB
   frD[32..63] ← (frB), преобразованное в 32-разрядное целое со знаком.
   Режим округления задается битом FPSCR[RN]. Биты frD[0..31] не определены.
   Если (frB) > $2^{31} - 1$, то frD[32..63] ← x'7FFFFFFF'.
   Если (frB) < -2^{31}, то frD[32..63] ← x'80000000'.
3) Floating-Point Convert to Integer Word with Round toward Zero
   fctiwz[.] frD,frB
   frD[32..63] ← (frB), преобразованное в 32-разрядное целое со знаком, с использованием режима округления в сторону нуля.
   Биты frD[0..31] не определены.
   Если (frB) > $2^{31} - 1$, то frD[32..63] ← x'7FFFFFFF'.
   Если (frB) < -2^{31}, то frD[32..63] ← x'80000000'.

4.4. Команды сравнения с плавающей запятой

Команды сравнения устанавливают соответствующий результату сравнения бит заданного в команде поля CR и очищают три других бита этого поля. Так же заполняется поле кода состояния с плавающей запятой FPSCR[16..19].

Интерпретация битов полей CR и FPSCR[16..19] приведена в таблице 2.

1) Floating-Point Compare Unordered
   fcmpu crfD,frA,frB
   Результат неупорядоченного сравнения (frA) и (frB) помещается в поле CR, заданное операндом crfD и в FPSCR[16..19].
2) Floating-Point Compare Ordered
   fcmpo crfD,frA,frB
   Результат упорядоченного сравнения (frA) и (frB) помещается в поле CR, заданное операндом crfD и в FPSCR[16..19].

4.5. Команды управления FPSCR

Все команды с плавающей запятой, инициированные ранее, завершают выполнение до начала выполнения команды управления FPSCR, и пока выполнение команды управления FPSCR не завершено, ни одна команда с плавающей запятой не может начать выполнение.

Команды управления FPSCR не влияют на выполнение команд с плавающей запятой, занимающихся обменом с памятью.

1) Move from FPSCR
   mffs[.] frD
   frD[32..63] ← (FPSCR)
   frD[0..31] не определены.
2) Move to Condition Register from FPSCR
   mcrfs crfD,crfS
   Содержимое поля FPSCR, заданного операндом crfS, копируется в поле CR, заданное операндом crfD. Все копируемые биты исключений в FPSCR обнуляются.
3) Move to FPSCR Field Immediate
   mtfsfi[.] crfD,IMM
   В поле FPSCR, заданное операндом crfD, копируется (IMM). Копирование в поле FPSCR[0..3] осуществляется по особым правилам.
4) Move to FPSCR Fields
   mtfsf[.] FM,frB
   frB[32..63] копируется в FPSCR под управлением маски FM. 8 битов маски соответствуют восьми 4-битовым полям FPSCR. Копирование в поле FPSCR[0..3] осуществляется по особым правилам.
5) Move to FPSCR Bit 0
   mtfsb0[.] crbD
   Указанный операндом crbD бит FPSCR обнуляется. Биты 1 и 2 FPSCR не могут быть явно обнулены.

Таблица 2

Интерпретация битов поля CR

Бит	Имя	Описание
0	FL	(frA) < (frB)
1	FG	(frA) > (frB)
2	FE	(frA) = (frB)
3	FU	(frA) ? (frB) (не упорядочено)

6) Move to FPSCR Bit 1
mtfsb1[.] crbD
Указанный операндом crbD бит FPSCR устанавливается в 1. Биты 1 и 2 FPSCR не могут быть явно установлены.

5. Команды обмена с памятью

Команды обмена с памятью делятся на следующие группы:
- целые команды чтения из памяти;
- целые команды записи в память;
- целые команды чтения и записи с обращением байтов;
- целые множественные команды чтения и записи;
- команды чтения из памяти с плавающей запятой;
- команды записи в память с плавающей запятой;
- команды пересылки с плавающей запятой;
- команды синхронизации памяти.

В целых командах обмена с памятью используются *три режима адресации.*

Режим косвенного регистра с непосредственным индексом:
адрес = (rA|0) + d.
Режим косвенного регистра с индексным регистром:
адрес = (rA|0) + (rB).
Режим косвенного регистра:
адрес = (rA|0).

5.1. Целые команды чтения из памяти

1) Load Byte and Zero
lbz rD,d(rA)
EA = (rA|0) + d
rD[24..31] ← байт(EA)
rD[0..23] ← x'000000'
2) Load Byte and Zero Indexed
lbzx rD,rA,rB
EA = (rA|0) + (rB)
rD[24..31] ← байт(EA)
rD[0..23] ← x'000000'
3) Load Byte and Zero with Update
lbzu rD,d(rA)
EA = (rA|0) + d
rD[24..31] ← байт(EA)
rD[0..23] ← x'000000'
Если not[(rA = 0) | (rA = rD)], то rA ← EA
601 допускает случаи (rA = 0) и (rA = rD), хотя архитектура PowerPC рассматривает их как ошибочные.
4) Load Byte and Zero with Update Indexed
lbzux rD,rA,rB
EA = (rA|0) + (rB)
rD[24..31] ← байт(EA)
rD[0..23] ← x'000000'
Если not[(rA = 0) | (rA = rD)], то rA ← EA
601 допускает случаи (rA = 0) и (rA = rD), хотя архитектура PowerPC рассматривает их как ошибочные.
5) Load Half Word and Zero
lhz rD,d(rA)
EA = (rA|0) + d
rD[16..31] ← полуслово(EA)
rD[0..15] ← x'0000'
6) Load Half Word and Zero Indexed
lhzx rD,rA,rB
EA = (rA|0) + (rB)
rD[16..31] ← полуслово(EA)
rD[0..15] ← x'0000'
7) Load Half Word and Zero with Update
lhzu rD,d(rA)
EA = (rA|0) + d
rD[16..31] ← полуслово(EA)
rD[0..15] ← x'0000'
Если not[(rA = 0) | (rA = rD)], то rA ← EA
601 допускает случаи (rA = 0) и (rA = rD), хотя архитектура PowerPC рассматривает их как ошибочные.
8) Load Half Word and Zero with Update Indexed
lhzux rD,rA,rB
EA = (rA|0) + (rB)
rD[16..31] ← полуслово(EA)
rD[0..15] ← x'0000'
Если not[(rA = 0) | (rA = rD)], то rA ← EA
601 допускает случаи (rA = 0) и (rA = rD), хотя архитектура PowerPC рассматривает их как ошибочные.
9) Load Half Word Algebraic
lha rD,d(rA)
EA = (rA|0) + d
rD[16..31] ← полуслово(EA)
rD[0],rD[1],..,rD[15] ← rD[16]

Начало на стр. 8

Таблица 1

Результаты тестирования 83 МГц Pentium OverDrive

Тест	Подраздел теста	386/387 DX-40	486DX2-66	83 МГц Pentium OverDrive	Примечания
Byte	CPU matrix	1,5	0,62	0,38	Время выполнения: чем меньше, тем лучше
	string	10,0;12,5;5,02	12,67;10,32;7,95	9,52;7,14;4,78	
	sieve	9,54	3,1	1,76	
	sort	7,16	2,39	1,32	
	FPU math	2,93	0,64	0,4	
	sin(x)	1,05	0,39	0,18	
	exp(x)	1,05	0,31	0,13	
PCLabs56	Instruxion mix	1,34	0,63	0,48	Время выполнения: чем меньше, тем лучше
	Integer add loop	0,3	0,09	0,05	
	Integer multiply loop	0,2	0,09	0,04	
	String sort and move	0,44	0,17	0,14	
	Prime number sieve	0,2	0,06	0,03	
	Floating point mix	3,19	1,37	1,05	
Wintach	Word processing	1,84	5,03	6,53	Производительность: чем больше, тем лучше
	CAD/Draw	2,23	4,30	6,19	
	Spreadsheet	1,57	4,51	5,34	
	Paint	1,92	5,51	6,92	
S1	Photoshop + AmiPro	-	39,8	25,8	Время выполнения: чем меньше, тем лучше
S2	Excel	-	28,0	23,0	
S3	Lotus 1-2-3	-	24,0	21,0	

10) Load Half Word Algebraic Indexed
lhax rD,rA,rB
EA = (rA|0) + (rB)
rD[16..31] ← полуслово(EA)
rD[0],rD[1],...,rD[15] ← rD[16]
11) Load Half Word Algebraic with Update
lhau rD,d(rA)
EA = (rA|0) + d
rD[16..31] ← полуслово(EA)
rD[0],rD[1],...,rD[15] ← rD[16]
Если not[(rA = 0) | (rA = rD)], то rA ← EA
601 допускает случаи (rA = 0) и (rA = rD), хотя архитектура PowerPC рассматривает их как ошибочные.
12) Load Half Word Algebraic with Update Indexed
lhaux rD,rA,rB
EA = (rA|0) + (rB)
rD[16..31] ← полуслово(EA)
rD[0],rD[1],...,rD[15] ← rD[16]
Если not[(rA = 0) | (rA = rD)], то rA ← EA
601 допускает случаи (rA = 0) и (rA = rD), хотя архитектура PowerPC рассматривает их как ошибочные.
13) Load Word and Zero
lwz rD,d(rA)
EA = (rA|0) + d
rD[0..31] ← слово(EA)
14) Load Word and Zero Indexed
lwzx rD,rA,rB
EA = (rA|0) + (rB)
rD[0..31] ← слово(EA)
15) Load Word and Zero with Update
lwzu rD,d(rA)
EA = (rA|0) + d
rD[0..31] ← слово(EA)
Если not[(rA = 0) | (rA = rD)], то rA ← EA
601 допускает случаи (rA = 0) и (rA = rD), хотя архитектура PowerPC рассматривает их как ошибочные.
16) Load Word and Zero with Update Indexed
lwzux rD,rA,rB
EA = (rA|0) + (rB)
rD[0..31] ← слово(EA)
Если not[(rA = 0) | (rA = rD)], то rA ← EA
601 допускает случаи (rA = 0) и (rA = rD), хотя архитектура PowerPC рассматривает их как ошибочные.

5.2. Целые команды записи в память

1) Store Byte
stb rS,d(rA)
EA = (rA|0) + d
байт(EA) ← rS[24..31]
2) Store Byte Indexed
stbx rS,rA,rB
EA = (rA|0) + (rB)
байт(EA) ← rS[24..31]
3) Store Byte with Update
stbu rS,d(rA)
EA = (rA|0) + d
байт(EA) ← rS[24..31]
Если not(rA = 0), то rA ← EA
601 допускает случай (rA = 0), хотя архитектура PowerPC рассматривает его как ошибочный.
4) Store Byte with Update Indexed
stbux rS,rA,rB
EA = (rA|0) + (rB)
байт(EA) ← rS[24..31]
Если not(rA = 0), то rA ← EA
601 допускает случай (rA = 0), хотя архитектура PowerPC рассматривает его как ошибочный.
5) Store Half Word
sth rS,d(rA)
EA = (rA|0) + d
полуслово(EA) ← rS[16..31]
6) Store Half Word Indexed
sthx rS,rA,rB
EA = (rA|0) + (rB)
полуслово(EA) ← rS[16..31]
7) Store Half Word with Update
sthu rS,d(rA)
EA = (rA|0) + d
полуслово(EA) ← rS[16..31]
Если not(rA = 0), то rA ← EA
601 допускает случай (rA = 0), хотя архитектура PowerPC рассматривает его как ошибочный.
8) Store Half Word with Update Indexed
sthux rS,rA,rB
EA = (rA|0) + (rB)
полуслово(EA) ← rS[16..31]
Если not(rA = 0), то rA ← EA
601 допускает случай (rA = 0), хотя архитектура PowerPC рассматривает его как ошибочный.
9) Store Word
stw rS,d(rA)
EA = (rA|0) + d
слово(EA) ← (rS)
10) Store Word Indexed
stwx rS,rA,rB
EA = (rA|0) + (rB)
слово(EA) ← (rS)
11) Store Word with Update
stwu rS,d(rA)
EA = (rA|0) + d
слово(EA) ← (rS)
Если not(rA = 0), то rA ← EA
601 допускает случай (rA = 0), хотя архитектура PowerPC рассматривает его как ошибочный.
12) Store Word with Update Indexed
stwux rS,rA,rB
EA = (rA|0) + (rB)
слово(EA) ← (rS)
Если not(rA = 0), то rA ← EA
601 допускает случай (rA = 0), хотя архитектура PowerPC рассматривает его как ошибочный.

5.3. Целые команды чтения и записи с обращением байтов

1) Load Half Word Byte-Reverse Indexed
lhbrx rD,rA,rB
EA = (rA|0) + (rB)
rD[24..31] ← полуслово(EA)[0..7]
rD[16..23] ← полуслово(EA)[8..15]
rD[0..15] ← x'0000'
2) Load Word Byte-Reverse Indexed
lwbrx rD,rA,rB
EA = (rA|0) + (rB)
rD[24..31] ← слово(EA)[0..7]
rD[16..23] ← слово(EA)[8..15]
rD[8..15] ← слово(EA)[16..23]
rD[0..7] ← слово(EA)[24..31]
3) Store Half Word Byte-Reverse Indexed
sthbrx rS,rA,rB
EA = (rA|0) + (rB)
полуслово(EA)[0..7] ← rS[24..31]
полуслово(EA)[8..15] ← rS[16..23]
4) Store Word Byte-Reverse Indexed
stwbrx rS,rA,rB
EA = (rA|0) + (rB)
слово(EA)[0..7] ← rS[24..31]
слово(EA)[8..15] ← rS[16..23]
слово(EA)[16..23] ← rS[8..15]
слово(EA)[24..31] ← rS[0..7]

5.4. Целые множественные команды чтения и записи

1) Load Multiple Word
lmw rD,d(rA)

EA = (rA|0) + d
GPR[rD .. 31] ← (32 — rD) последовательных слова, начиная с адреса EA
Если rD ≤ rA ≤ 31, то в 601 содержимое rA не меняется, хотя архитектура PowerPC считает такой случай ошибочным.
Если EA не кратен 4, то при переходе через границу страницы может случиться исключительная ситуация по выравниванию.

2) Store Multiple Word
stmw rS,d(rA)
EA = (rA|0) + d
(32 — rS) последовательных слова, начиная с адреса
EA ← GPR[rS .. 31]
Если EA не кратен 4, то при переходе через границу страницы может случиться исключительная ситуация по выравниванию.

3) Load String Word Immediate
lswi rD,rA,NB
EA = (rA|0)
Если NB = 0, то n = 32, иначе n = NB;
nr = CEIL(n/4).
GPR[rD .. rD+nr-1] ← n последовательных байтов, начиная с адреса EA
Незаполненные младшие байты GPR[rD+nr-1] заполняются нулями.
Если rD ≤ rA ≤ rD+nr-1, то содержимое rA не меняется.

4) Load String Word Indexed
lswx rD,rA,rB
EA = (rA|0) + (rB)
n = XER[25..31]; nr = CEIL(n/4).
Если n = 0, то (rD) не определено.
Если n > 0, то GPR[rD .. rD+nr-1] ← n последовательных байтов, начиная с адреса EA
Незаполненные младшие байты GPR[rD+nr-1] заполняются нулями.
Если rD ≤ rA ≤ rD+nr-1, то содержимое rA не меняется.
Если rD ≤ rB ≤ rD+nr-1, то содержимое rB не меняется.

5) Store String Word Immediate
stswi rS,rA,NB
EA = (rA|0)
Если NB = 0, то n = 32, иначе n = NB; nr = CEIL(n/4).
n последовательных байтов, начиная с адреса
EA ← GPR[rS .. rS+nr-1]

6) Store String Word Indexed
stswx rS,rA,rB
EA = (rA|0) + (rB)
n = XER[25..31]; nr = CEIL(n/4).
n последовательных байтов, начиная с адреса
EA ← GPR[rS .. rS+nr-1]

5.5. Команды чтения из памяти с плавающей запятой

1) Load Floating-Point Single-Precision
lfs frD,d(rA)
EA = (rA|0) + d
Слово(EA) рассматривается как число с плавающей запятой одинарной точности, преобразуется к двойной точности и помещается в frD.

2) Load Floating-Point Single-Precision Indexed
lfsx frD,rA,rB
EA = (rA|0) + (rB)
Слово(EA) рассматривается как число с плавающей запятой одинарной точности, преобразуется к двойной точности и помещается в frD.

3) Load Floating-Point Single-Precision with Update
lfsu frD,d(rA)
EA = (rA|0) + d
Слово(EA) рассматривается как число с плавающей запятой одинарной точности, преобразуется к двойной точности и помещается в frD.
Если not(rA = 0), то rA ← EA.
601 допускает случай (rA = 0), хотя архитектура PowerPC рассматривает его как ошибочный.

4) Load Floating-Point Single-Precision with Update Indexed
lfsux frD,rA,rB
EA = (rA|0) + (rB)
Слово(EA) рассматривается как число с плавающей запятой одинарной точности, преобразуется к двойной точности и помещается в frD.
Если not(rA = 0), то rA ← EA.
601 допускает случай (rA = 0), хотя архитектура PowerPC рассматривает его как ошибочный.

5) Load Floating-Point Double-Precision
lfd frD,d(rA)
EA = (rA|0) + d
frD ← двойное слово(EA)

6) Load Floating-Point Double-Precision Indexed
lfdx frD,rA,rB
EA = (rA|0) + (rB)
frD ← двойное слово(EA)

7) Load Floating-Point Double-Precision with Update
lfdu frD,d(rA)
EA = (rA|0) + d
frD ← двойное слово(EA)
Если not(rA = 0), то rA ← EA.
601 допускает случай (rA = 0), хотя архитектура PowerPC рассматривает его как ошибочный.

8) Load Floating-Point Double-Precision with Update Indexed
lfdux frD,rA,rB
EA = (rA|0) + (rB)
frD ← двойное слово(EA)
Если not(rA = 0), то rA ← EA.
601 допускает случай (rA = 0), хотя архитектура PowerPC рассматривает его как ошибочный.

5.6. Команды записи в память с плавающей запятой

1) Store Floating-Point Single-Precision
stfs frS,d(rA)
EA = (rA|0) + d
Содержимое frS преобразуется к одинарной точности и помещается в слово(EA).

2) Store Floating-Point Single-Precision Indexed
stfsx frS,rA,rB
EA = (rA|0) + (rB)
Содержимое frS преобразуется к одинарной точности и помещается в слово(EA).

3) Store Floating-Point Single-Precision with Update
stfsu frS,d(rA)
EA = (rA|0) + d
Содержимое frS преобразуется к одинарной точности и помещается в слово(EA).
Если not(rA = 0), то rA ← EA.
601 допускает случай (rA = 0), хотя архитектура PowerPC рассматривает его как ошибочный.

4) Store Floating-Point Single-Precision with Update Indexed
stfsux frS,rA,rB
EA = (rA|0) + (rB)
Содержимое frS преобразуется к одинарной точности и помещается в слово(EA).
Если not(rA = 0), то rA ← EA.
601 допускает случай (rA = 0), хотя архитектура PowerPC рассматривает его как ошибочный.

5) Store Floating-Point Double-Precision
stfd frS,d(rA)
EA = (rA|0) + d
двойное слово(EA) ← (frS)

6) Store Floating-Point Double-Precision Indexed
stfdx frS,rA,rB
EA = (rA|0) + (rB)
двойное слово(EA) ← (frS)

7) Store Floating-Point Double-Precision with Update
stfdu frS,d(rA)

EA = (rA|0) + d
двойное слово(EA) ← (frS)
Если not(rA = 0), то rA ← EA.
601 допускает случай (rA = 0), хотя архитектура PowerPC рассматривает его как ошибочный.
8) Store Floating-Point Double-Precision with Update Indexed
stfdux frS,rA,rB
EA = (rA|0) + (rB)
двойное слово(EA) ← (frS)
Если not(rA = 0), то rA ← EA.
601 допускает случай (rA = 0), хотя архитектура PowerPC рассматривает его как ошибочный.

5.7. Команды пересылки с плавающей запятой

1) Floating-Point Move Register
fmr[.] frD,frB
frD ← (frB)
2) Floating-Point Negate
fneg[.] frD,frB
frD ← — (frB)
3) Floating-Point Absolute Value
fabs[.] frD,frB
frD ← |(frB)|
4) Floating-Point Negative Absolute Value
fnabs[.] frD,frB
frD ← — |(frB)|

5.8. Команды синхронизации памяти

Эти команды управляют порядком завершения инициированных ранее команд обмена с памятью в случаях, когда необходимо организовать правильную обработку асинхронных событий или взаимодействие с другими процессорами.

1) Enforce In-Order Execution of I/O
eieio
Упорядочивает действие команд чтения из памяти и записи в память, выполняемых данным процессором. Все обращения к основной памяти, инициированные ранее процессором, завершаются, и до момента их завершения процессор не может инициировать ни одного нового обращения к основной памяти. Команда упорядочивает операции чтения и записи для памяти с запрещенной записью в кэш, а также операции записи для кэша с прямой записью.
2) Instruction Synchronize
isync
Эта команда ждет завершения выполнения всех инициированных ранее команд, затем убирает выбранные к этому моменту из очереди на выполнение команды, вследствие чего выборка последующих команд и их выполнение происходят в контексте, установленном командами, предшествующими isync.
3) Load Word and Reserve Indexed
lwarx rD,rA,rB
EA = (rA|0) + (rB)
rD ← слово(EA)
EA резервируется для последующей команды stwcx. EA должен быть кратен 4.
4) Store Word Conditional Indexed
stwcx. rS,rA,rB
EA = (rA|0) + (rB)
Если есть резервирование, то слово(EA) ← rS и резервирование отменяется. Если резервирования нет, то команда завершается, не меняя содержимого памяти или кэша.
Бит EQ поля CR0 в регистре CR устанавливается в 1, если запись в память произошла.
Пара команд lwarx и stwcx может применяться для организации в памяти семафоров.
5) Synchronize
sync
Все команды, инициированные ранее процессором, завершаются, и до момента их завершения процессор не может инициировать ни одной новой команды.
В 601 команды sync и eieio выполняются одинаково.

6. Команды, изменяющие порядок выполнения программы

6.1. Команды перехода

601 предполагает, что адреса команд выровнены по границе слова, поэтому два младших бита команды используются как расширение кода операции.

Бит 30 (AA) = 0 — в команде задано смещение;
AA = 1 — в команде задан абсолютный адрес.

Бит 31 (LK) = 1 — адрес команды, следующей за командой перехода, будет помещен в LR; LK = 0 — команда не влияет на LR.

Операнд BO описывает условия, при которых осуществляется переход. Возможные значения операнда BO приведены в таблице 3 (бит z должен быть равен 0; y=1 — предсказывается, что переход произойдет для всех команд, кроме bcx с отрицательным смещением, y=0 — наоборот).

1) Branch
b LI
EA ← (адрес текущей команды) + (LI || b'00')
Безусловный переход по EA.
2) Branch Absolute
ba LI
EA ← LI || b'00'
Безусловный переход по EA.
3) Branch then Link
bl LI
EA ← (адрес текущей команды) + (LI || b'00')
Безусловный переход по EA. LR ← (адрес следующей команды).
4) Branch Absolute then Link
bla LI
EA ← LI || b'00'
Безусловный переход по EA. LR ← (адрес следующей команды).
5) Branch Conditional
bc BO,BI,BD
EA ← (адрес текущей команды) + (BD || b'00')
Условный переход по EA.
6) Branch Conditional Absolute
bca BO,BI,BD
EA ← BD || b'00'
Условный переход по EA.
7) Branch Conditional then Link
bcl BO,BI,BD
EA ← (адрес текущей команды) + (BD || b'00')
Условный переход по EA. LR ← (адрес следующей команды).
8) Branch Conditional Absolute then Link
bcla BO,BI,BD
EA ← BD || b'00'
Условный переход по EA. LR ← (адрес следующей команды).
9) Branch Conditional to Link Register
bclr BO,BI
EA ← (LR)
Условный переход по EA.
10) Branch Conditional to Link Register then Link

Таблица 3

Значения операнда BO

BO	Описание
0000y	CTR = CTR — 1; переход, если not(CTR = 0) & (условие = FALSE)
0001y	CTR = CTR — 1; переход, если (CTR = 0) & (условие = FALSE)
001zy	Переход, если (условие = FALSE)
0100y	CTR = CTR — 1; переход, если not(CTR = 0) & (условие = TRUE)
0101y	CTR = CTR — 1; переход, если (CTR = 0) & (условие = TRUE)
011zy	Переход, если (условие = TRUE)
1z00y	CTR = CTR — 1; переход, если not(CTR = 0)
1z01y	CTR = CTR — 1; переход, если (CTR = 0)
1z1zz	Безусловный переход

bclrl BO,BI
EA ← (LR)
Условный переход по EA. LR ← (адрес следующей команды).
11) Branch Conditional to Count Register
bcctr BO,BI
EA ← (CTR)
Условный переход по EA.
12) Branch Conditional to Count Register then Link
bcctrl BO,BI
EA ← (CTR)
Условный переход по EA. LR ← (адрес следующей команды).

Существует несколько десятков удобных мнемонических обозначений для различных вариантов команд перехода, все они приведены в руководстве [2].

6.2. Команды для регистра условия

Эти команды изменяют содержимое регистра условия CR. Они выполняются целочисленным устройством (IU), однако приведены в данном разделе, поскольку обычно они используются вместе с командами перехода.

1) Condition Register AND
crand crbD,crbA,crbB
CR[crbD] ← CR[crbA] & CR[crbB]
2) Condition Register OR
cror crbD,crbA,crbB
CR[crbD] ← CR[crbA] | CR[crbB]
3) Condition Register XOR
crxor crbD,crbA,crbB
CR[crbD] ← CR[crbA] xor CR[crbB]
4) Condition Register NAND
crnand crbD,crbA,crbB
CR[crbD] ← not(CR[crbA] & CR[crbB])
5) Condition Register NOR
crnor crbD,crbA,crbB
CR[crbD] ← not(CR[crbA] | CR[crbB])
6) Condition Register Equivalent
creqv crbD,crbA,crbB
CR[crbD] ← not(CR[crbA] xor CR[crbB])
7) Condition Register AND with Complement
crandc crbD,crbA,crbB
CR[crbD] ← CR[crbA] & not(CR[crbB])
8) Condition Register OR with Complement
crorc crbD,crbA,crbB
CR[crbD] ← CR[crbA] | not(CR[crbB])
9) Move Condition Register Field
mcrf crfD,crfS
CR[4*crfD .. 4*crfD+3] ← CR[4*crfS .. 4*crfS+3]

6.3. Команды системной связи

1) System Call
sc
SPR0 ← адрес следующей команды
SPR1[0..15] — не определены. SPR1[16..31] ← MSR[16..31]. Затем генерируется исключительная ситуация системного вызова.
2) Return from Interrupt
rfi
MSR[16..31] ← SPR1[16..31]. Следующая команда выбирается по адресу (SPR0[0..29] ‖ b'00'). Это команда уровня супервизора.

6.4. Команды проверки условия

Операнд TO — поле, задающее условие (см. таблицу 4).

1) Trap Word Immediate
twi TO,rA,SIMM
(rA) сравнивается с SIMM. Если какой-нибудь из битов TO установлен в 1 и соответствующее ему условие выполнено, то вызывается системный обработчик (trap handler).
2) Trap Word
tw TO,rA,rB
(rA) сравнивается с (rB). Если какой-нибудь из битов TO установлен в 1 и соответствующее ему условие выполнено, то вызывается системный обработчик (trap handler).

Предусмотрено большое количество мнемоник для различных вариантов команд проверки условия. Все они приведены в [2].

7. Команды управления процессором

1) Move to Condition Register Fields
mtcrf CRM,rS
CR ← (rS) под управлением маски CRM.
2) Move to Condition Register from XER
mcrxr crfD
CR[4*crfD .. 4*crfD+3] ← XER[0..3]. XER[0..3] ← b'0000'.
3) Move from Condition Register
mfcr rD
rD ← (CR)
4) Move to Machine State Register
mtmsr rS
MSR ← (rS)
Это команда уровня супервизора.
5) Move from Machine State Register
mfmsr rD
rD ← (MSR)
Это команда уровня супервизора.
6) Move to Special Purpose Register
mtspr SPR,rS
Регистр, заданный полем SPR ← (rS)
7) Move from Special Purpose Register
mfspr rD,SPR
rD ← (регистр, заданный полем SPR)

В [2] приведены упрощенные мнемоники для команд обмена с регистрами специального назначения.

8. Команды управления памятью

8.1. Команды управления кэш-памятью

В этом разделе термин «блок» обозначает сектор кэш-памяти размером 32 байта.

1) Data Cache Block Invalidate
dcbi rA,rB
EA ← (rA|0) + (rB)
Команда уровня супервизора. Объявляет недействительным блок в кэше, содержащий байт с адресом EA. Действие команды может распространяться как на текущий процессор, так и на все процессоры в системе. Подробное описание см. [2].
2) Data Cache Block Touch
dcbt rA,rB
EA ← (rA|0) + (rB)
В кэш загружается блок, содержащий байт с адресом EA. Цель команды — последующая быстрая загрузка данных в регистры. Подробное описание см. [2].
3) Data Cache Block Touch for Store
dcbtst rA,rB
EA ← (rA|0) + (rB)
В кэш загружается блок, содержащий байт с адресом EA. Цель команды — последующая быстрая запись данных в память. В 601 эта команда выполняется точно так же, как и dcbt. Подробное описание см.[2].
4) Data Cache Block Set to Zero

Таблица 4

Значения битов операнда TO

Бит	Условие
0	Меньше чем
1	Больше чем
2	Равно
3	Логически меньше чем
4	Логически больше чем

dcbz rA,rB
EA ← (rA|0) + (rB)
Если блок, содержащий байт с адресом EA, находится в кэше, то все байты этого блока обнуляются. Подробное описание см. [2].

5) Data Cache Block Store
dcbst rA,rB
EA ← (rA|0) + (rB)
Если блок, содержащий байт с адресом EA, находится в кэше любого из процессоров системы, включен режим поддержания кэш-соответствия и содержимое блока было изменено, то инициируется его запись в основную память. Подробное описание см. [2].

6) Data Cache Block Flush
dcbf rA,rB
EA ← (rA|0) + (rB)
Копирует блок, содержащий байт с адресом EA и находящийся в кэше одного из процессоров системы, в основную память, после чего объявляет этот блок недействительным. Подробное описание см. [2].

8.2. Команды чтения и записи для регистров сегмента

Команды работы с регистрами сегмента — команды уровня супервизора.

1) Move to Segment Register
mtsr SR,rS
SR ← (rS)

2) Move to Segment Register Indirect
mtsrin rS,rB
Регистр сегмента с номером rB[0..3] ← (rS)

3) Move from Segment Register
mfsr rD,SR
rD ← (SR)

4) Move from Segment Register Indirect
mfsrin rD,rB
rD ← (Регистр сегмента с номером rB[0..3])

8.3. Команда управления буфером с таблицей трансляции адресов (TLB)

1) Translation Lookaside Buffer Invalidate Entry
tlbie rB
EA ← (rB)
Если в TLB содержится строка, соответствующая EA, то эта строка удаляется из TLB. Подробное описание см. [2].

9. Команды внешнего управления

Эти команды позволяют программам пользователя взаимодействовать с устройствами специального назначения.

1) External Control Input Word Indexed
eciwx rD,rA,rB
EA ← (rA|0) + (rB)
EA должен указывать на границу слова. Если бит регистра внешнего доступа EAR[0] установлен в 1, то в обход кэша посылается запрос на чтение по физическому адресу, соответствующему EA, на устройство, заданное битами EAR[28..31]. Полученное по запросу слово помещается в rD. Подробное описание см. [2].

2) External Control Output Word Indexed
ecowx rS,rA,rB
EA ← (rA|0) + (rB)
EA должен указывать на границу слова. Если бит регистра внешнего доступа EAR[0] установлен в 1, то (rS) посылается в обход кэша для записи по физическому адресу, соответствующему EA, на устройство, заданное битами EAR[28..31]. Подробное описание см. [2]. ■

Автор благодарен сотруднику фирмы RUI Apple Computer Олегу Котику за содействие при подготовке статьи.

Литература

1. **Д. Бройтман. Микропроцессор PowerPC 601// Монитор. — 1994. — № 4.**

2. **PowerPC 601 RISC Microprocessor User's Manual. — USA: Motorola, 1994.**

Условные обозначения

[.] — к имени команды может быть добавлен суффикс «.».
[o] — к имени команды может быть добавлен суффикс «o».
← — присваивание.
|| — конкатенация.
not — логическая операция НЕ.
& — логическая операция И.
| — логическая операция ИЛИ.
xor — логическая операция ИСКЛЮЧАЮЩЕЕ ИЛИ.
x'nnnn' — число в шестнадцатеричном формате.
b'nn' — число в двоичном формате.
rA — биты 11-15 команды — номер первого из регистров-источников.
(rA) — содержимое регистра rA.
(rA|0) — содержимое регистра rA, если поле rA содержит 1-31; 0 — если поле rA содержит 0.
rB — биты 16-20 команды — номер второго из регистров-источников.
(rB) — содержимое регистра rB.
rD — биты 6-10 команды — номер регистра-результата.
rS — биты 6-10 команды — номер регистра-источника.
frA — биты 11-15 команды — номер первого из регистров-источников с плавающей запятой.
frB — биты 16-20 команды — номер второго из регистров-источников с плавающей запятой.
frC — биты 21-25 команды — номер третьего из регистров-источников с плавающей запятой.
frD — биты 6-10 команды — номер регистра-результата с плавающей запятой.
frS — биты 6-10 команды — номер регистра-источника с плавающей запятой.
SIMM — биты 16-31 команды — непосредственный операнд (16-битовое целое со знаком).
UIMM — биты 16-31 команды — непосредственный операнд (16-битовое целое без знака).
IMM — биты 16-19 команды — непосредственный операнд, который должен быть помещен в поле FPSCR.
crbA — биты 11-15 команды — задает бит-источник в CR.
crbB — биты 16-20 команды — задает бит-источник в CR.
crbD — биты 6-10 команды — задает бит в CR или FPSCR в качестве результата.
crfD — биты 6-8 команды — задает поле CR или поле FPSCR в качестве результата.
crfS — биты 11-13 команды — задает поле CR или поле FPSCR в качестве источника.
CRM — биты 12-19 команды — маска, задающая поля в CR.
FM — биты 7-14 команды — маска, задающая поля FPSCR.
d — биты 16-31 команды — непосредственный индекс (16-битовое целое со знаком).
EA — адрес, по которому происходит обращение к памяти.
LI — биты 6-29 команды — смещение или абсолютный адрес.
BD — биты 16-29 команды — смещение или абсолютный адрес.
BO — биты 6-10 команды — описывает условия, при которых осуществляется переход.
BI — биты 11-15 команды — задает бит-условие в CR.
TO — биты 6-10 команды — поле, задающее условие.
SPR — биты 11-20 команды — операнд, задающий регистр специального назначения. Кодировку поля SPR см. в [2].
SR — биты 12-15 команды — номер регистра сегмента.

Nexus

Москва,
Кузнецкий мост 21/5
921-36-70, 928-23-67

OKI
People to People Technology

Сделайте правильный выбор

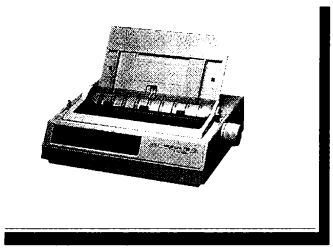


Таблица сравнительных характеристик

	ML 321/320	EPSON FX 1050/850
Скорость печати	360 зн/сек	290 зн/сек
Плотность зн/дюйм	10,12,15,17,20	10,12,17,20
Число копий	Ориг+3	Ориг+3
Буфер	15 кб	8 кб
Печать графики	в две стороны	в одну сторону
Надежность головки	200 млн.симв.	100 млн.симв.

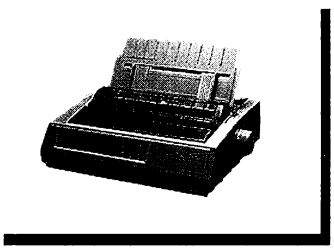


Таблица сравнительных характеристик

	ML 521/520	EPSON FX 1170/870
Скорость печати	433 зн/сек	380 зн/сек
Плотность зн/дюйм	10,12,15,17,20	10,12,17,20
Число копий	Ориг+5	Ориг+3
Буфер	64 кб	24 кб
Печать штрих-кодов	да	нет
Надежность головки	200 млн.симв.	100 млн.симв.

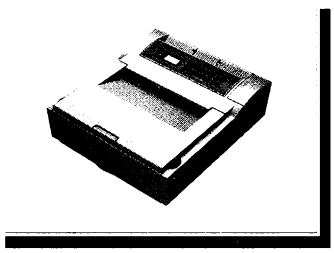


Таблица сравнительных характеристик

	OL 400e	OL 410e	HP 4L
Метод печати	LED	LED	LASER
Разрешение	300 dpi	300 (600 RET) dpi	300 (600 RET) dpi
Память (норм/макс.)	512kb/4.5 Mb	1/5 Mb	1/2 Mb
Ресурс работы барабана	20 000 копий	20 000 копий	3 000 копий
RS 232	есть	есть	нет
Команды	PCL 4	PCL 5	PCL 5

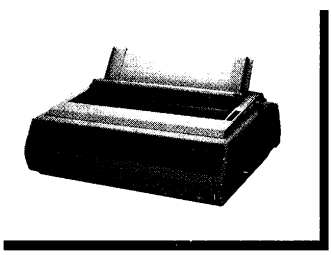
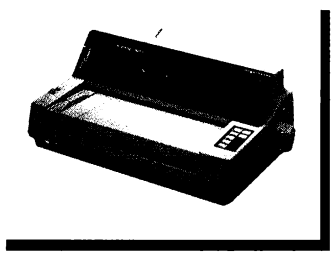


Таблица сравнительных характеристик

	ML 3410	EPSON DFX 5000
Скорость печати	550 зн/сек	533 зн/сек
Плотность зн/дюйм	10, 12, 15, 17, 20	10, 12
Число копий	Ориг+7	Ориг+5
Буфер	16 кб	3 кб
Ширина перф. бумаги	420 мм. (рос. стандарт)	406 мм.
Надежность принтера	8000 час.	6000 час.



ML 390FB

Печать без изгиба бумаги
на сберкнижках, карточках,
сертификатах акций, авиабилетах.
24 иголки, ширина документа до 30 см.

OKi
OK!
Приглашаются дилеры

Цены (розничные/дилерские)

ML 320 (узкий) 9 pin	380/350$
ML 321 (широкий) 9 pin	440/400$
ML 520 (узкий) 9 pin	460/420$
ML 521 (широкий) 9 pin	510/460$
ML 590 (узкий) 24 pin	590/530$
ML 591 (широкий) 24 pin	640/580$
ML 390FB (банк.) 24 pin	1100/1000$
ML 3410 9 pin	1250/1150$
OL 400e	650/600$
OL 410e	750/680$
OKIFAX OF 1000	1900/1750$

Факс на обычной бумаге

IBM OS/2 Warp — музыка атаки

А. Козленков

Рассказать об операционной системе OS/2 (правильное произношение — «оу-эс-ту», «ос-два», то есть операционная система второго поколения) на нескольких журнальных страницах достаточно трудно. Поэтому будет сделана попытка заставить звучать лишь главные темы, которые связаны с этим явлением компьютерной жизни. Сравнение с музыкой обусловлено тем, что разработка и развитие этой ОС связаны со стремлением уловить коллективные настроения общественности пользователей и разработчиков, не забывать о долгосрочных перспективах, тщательно прорабатывать каждое техническое решение на высоком теоретическом уровне. Уже эти простые соображения показывают, что анализ возможностей такой системы, изначально ориентированной на будущее, почти неизбежно оказывается неполным, поэтому наилучший способ познакомиться с OS/2 — использовать ее.

Исследовательский характер OS/2 и нечеткий маркетинг фирмы IBM долгое время создавали у пользователей и аналитиков ощущение, что проект «витает в облаках». Важно подчеркнуть, что именно соотношение между адекватным решением сегодняшних задач, стоящих перед конкретным пользователем и разработчиком, и внесением новых, не в полной мере доведенных до промышленного уровня технологий, в системе OS/2 Warp значительно более сбалансировано, чем в предыдущих версиях. Говоря еще более конкретно, именно последовательный объектно-ориентированный подход позволил ликвидировать противостояние между требованием логической завершенности и непротиворечивости системы и наличием в ней самой возможностей роста, так чтобы новая версия не стала облицовкой фасада или полной перестройкой (намек на Windows-95 без специального умысла).

Прежде чем перейти к более детальной характеристике возможностей системы OS/2 Warp, обратим внимание, что это первая из ОС фирмы IBM, официально нацеленная на мелкий бизнес или индивидуального пользователя. В то же время наряду с версией, объявленной 11 октября 1994 года, в ближайшие месяцы появится много других вариаций системы, в частности, с ориентацией на полную интеграцию в локальные сети предприятия. Наконец, для справки надо отметить, что продано около 6 миллионов экземпляров OS/2 версии выше 2.0, факт, выводящий OS/2 на уровень продукта принципиально массового использования и являющийся для ее разработчиков как предметом гордости, так и большой ответственности.

Нестандартно о стандартном

Прежде всего отметим, что OS/2 — одна из первых, если не первая ОС, изначально построенная на многопоточном — (multiple

threads) принципе в рамках вытесняющей многозадачности (preemptive multitasking), дополненной системой приоритетов и прерыванием при ожидании ввода-вывода. Любой процесс системы, как и сама система, может состоять из нескольких параллельно выполняющихся ветвей, или потоков, причем

1) разделение времени процессора (или нескольких процессоров в многопроцессорной версии OS/2 SMP) система производит именно между отдельными потоками;
2) поток работает в том же пространстве глобальных данных и автоматически имеет те же внешние соединения, например открытые файлы, что и весь процесс;
3) разработчику это очень просто реализовать;
4) многопоточность в ядре и сервисе системы изначально ориентирует разработчика на эффективный подход в рамках архитектуры «клиент/сервер».

Очевидный пример — сервер баз данных при появлении запроса запускает дополнительный поток на его обработку, что не требует дорогостоящего запуска отдельного процесса, и немедленно переходит в состояние готовности для приема новых обращений. Для синхронизации и коммуникации потоков и процессов имеется развитая система семафоров, каналов, очередей, несколько вариантов общей памяти. В ближайшей перспективе система будет построена на основе микроядра в спецификации Mach 3, что обеспечит ее переносимость.

Начиная с версии 2.0 (1992 г.), система стала 32-битовой и требует процессора не ниже 386. Это означает, что работа с памятью чрезвычайно упрощается, а надежность возрастает, поскольку адресные пространства процессов взаимно защищены. Программы используют сегменты размером 4 гигабайта лишь для целей оптимизации ресурсов, например сегмент кода обычно помечается как загружаемый при вызове (LOADONCALL), а некоторые сегменты данных динамической библиотеки могут быть объявлены как общие для использующих ее процессов. Обмен адресного пространства с диском происходит прозрачно путем постраничного обмена (напомним, что страницей в процессорах выше 386 называется выровненная область размером 4 кб), в связи с этим никаких усилий или предосторожностей от разработчика не требуется. В процессе отладки программы при повторном запуске все адреса переменных остаются неизменными, что облегчает поиск ошибок, связанных с неверным обращением к памяти.

Динамические библиотеки, или библиотеки динамической компоновки (dynamic-link library, DLL) — неотъемлемое свойство OS/2 и Windows, совместное детище «Microsoft» и IBM. OS/2 Warp практически полностью состоит из файлов с расширением DLL. В отличие от разделяемых библиотек системы UNIX и DLL в Windows, DLL в OS/2 все более ассоциируется с объектом или системой взаимосвязанных объектов, обладающих кодом инициализации и выхода, предоставляющих процессу-клиенту не только реентерабельные точки входа, но и общие или индивидуальные для процесса переменные, иерархию классов. Для DLL характерна автоматическая регистрация процессов-клиентов, так что библиотека автоматически выгружается и деинициализируется, как только последний клиент завершает работу. Важность механизма динамической компоновки еще более возрастает в связи с реализацией в OS/2 системной объектной модели (SOM), о которой речь пойдет ниже.

Высокопроизводительная файловая система (HPFS) — неотъемлемый элемент ОС, обеспечивающий поддержку длинных имен файлов и так называемых расширенных атрибутов, используемых при работе графическим интерфейсом Workplace Shell. HPFS снижает уровень фрагментации диска и повышает устойчивость файловой системы к сбоям. Опыт показывает, что она наиболее важна при использовании клиентов и серверов системы NFS для обмена файлами с UNIX или OS/2 машинами (поставляется с пакетом TCP/IP для OS/2). Кроме того, поскольку на OS/2 почти полностью перенесен свободно распространяемый инструментарий GNU, разрабатывавшийся первоначально для платформы UNIX, длинные имена и заглавные и строчные буквы становятся удобными и необходимыми. В связи с этим важно отметить, что для OS/2 характерна высокая и возрастающая степень «приятия» UNIX-мира. Пакет TCP/IP для OS/2 обеспечивает полную интеграцию в сети TCP/IP, включая telnet, ftp и другие утилиты, поддерживает RPC, sockets, NFS как клиент, так и сервер, X-Window версии 5 как сервер, так и клиент, а также обеспечивает разработку распределенных приложений, а для X-Window не только Xlib, но и Xt, и OSF/Motif инструментарий.

Поддержка DOS и Windows

Лозунг «Лучший DOS, чем DOS, лучший Windows, чем Windows!» — в Warp в значительной степени верен. OS/2 Warp теперь автоматически определяет наилучшие из полусотни параметров настройки данного экземляра сеанса DOS, в котором будет выполняться какая-либо известная программа. Таковыми для OS/2 оказываются около 200 игр, включая «DOOM». OS/2 позволяет использовать разные драйверы и AUTOEXEC.BAT, до

16 Мб XMS-памяти, до 32 Мб EMS, 512 Мб DPMI, все это в каждом из сеансов DOS. Программы в старом стандарте VCPI не могут использоваться, но, начиная с Phar Lap 4.0, проблем не возникает. В противоположность некоторым сообщениям, AutoCAD 12 для DOS работает прекрасно после установки специального небольшого корректирующего (AutoCAD) модуля, бесплатно поставляемого фирмой «Autodesk».

В версиях Warp для Windows (теперь это называется «без Windows») система ставится на любые DOS и Windows 3.1, 3.11 или Windows for Workgroups 3.11, не изменяя ни DOS, ни Windows. Начиная с версии 2.1 OS/2, прекрасно работает OLE 2.0, а также Clipboard и DDE, последние даже с программами OS/2. Уже в OS/2 2.1 можно было использовать как стандартный, так и расширенный режим Windows. Система позволяет запустить несколько Windows одновременно, что улучшает изоляцию ненадежных приложений Windows. Версия OS/2 Warp хорошо работает с программами Win32s 1.0 и 1.1, единственная несовместимость сегодня — невозможность работы с драйверами VxD. Начиная с версии 2.0 OS/2, программы Windows работают как в полноэкранном режиме, полностью скрывающем графическую среду OS/2 — Workplace Shell, так и в окне, в бесшовном (seamless) режиме (рис. 1). При перемещении «мыши» между OS/2 и Windows она меняет цвет, кроме того, адекватно работает менеджер палитры при переходе фокуса ввода между окнами. OS/2 Warp позволяет установить режим предварительной загрузки основных компонентов Windows, при этом скорость запуска приложений становится одинаковой с Windows. Все программы DOS и Windows работают с данными и программами на высокопроизводительной файловой системе HPFS и могут быть на нее установлены. Если вам все-таки необходима «чистая» DOS, можно загрузить специальную версию DOS с дискеты или с образа на диске, не покидая OS/2 и сохраняя доступ к разделам HPFS, или воспользоваться средствами Dual Boot, которые перепишут загрузочный сектор диска и восстановят файлы конфигурации DOS, а также произведут обратные операции по желанию пользователя, или MultiBoot, что позволяет в разных главных разделах диска держать разные операционные системы, например DOS, Linux, NT, Coherent и так далее.

Workplace Shell — объектно-ориентированная рабочая среда

Уже в 1991 году было опубликовано предложение IBM о новом поколении *общего метода доступа пользователя* (Common User Access, CUA'91). Напомним, что MS-Windows 3.1 и OS/2 версий до 1.3 используют стандарт CUA'89. Несмотря на продуманность и естественность основных положений CUA'91, было ясно, что это долгосрочная программа, — в полном объеме она еще не реализована даже IBM. Проблема состояла в создании существенно объектно-ориентированной среды, ключевой особенностью которой стала бы ориентация не на приложения, то есть категории, принадлежащие компьютерной технологии и не имеющие никакого реального эквивалента в повседневной деятельности человека, работающего в производственном цикле любого рода, а на документо-ориентированные объекты интерфейса, по возможности имеющие непосредственными аналогами те конечные продукты, которые создаются, используются, хранятся, изменяются и уничтожаются в ходе деятельности частного лица или ограничизации. В соответствии с новейшими принципами, формулируемыми в последнее время, в частности, корпорацией «Taligent», создание систем объектов, образующих библиотеки классов, не может существенно изменить к лучшему ситуацию с программным обеспечением. Необходимо, чтобы такие библиотеки были оснащены собственным сценарием поведения; они называются структурами (framework).

С технической точки зрения, пользовательская (графическая) среда OS/2, называемая Workplace Shell, является такой структурой, состоящей из объектов, образующих иерархию, в соответствии с классификацией их свойств и возможностей. Последние обозначаются термином *«интерфейс»* (не следует путать с пользовательским интерфейсом, который является лишь подмножеством интерфейса объекта). В OS/2 Warp интерфейс объектов специфицирован в соответствии со стандартом *CORBA* (Common Object Request Broker Architecture), а реализация основана на *системной объектной модели версии 2 (SOM-2)*, дополненной *системой распределенной объектной обработки (распределенная системная объектная модель, DSOM)*. SOM предоставляет собой независимый от языка механизм создания классов и объектов, это означает, что программист и даже пользователь визуальной системы программирования, например VisPro REXX, может использовать, наращивать или изменять классы, созданные, скажем, на C++, в свою очередь, передавая написанные на нем классы модулям, написанным на различных языках. Пара SOM/DSOM поддерживает множественное наследование, динамическое нахождение методов, модификацию их в ходе работы, динамически порождаемый вызов методов (DII), дистанцион-

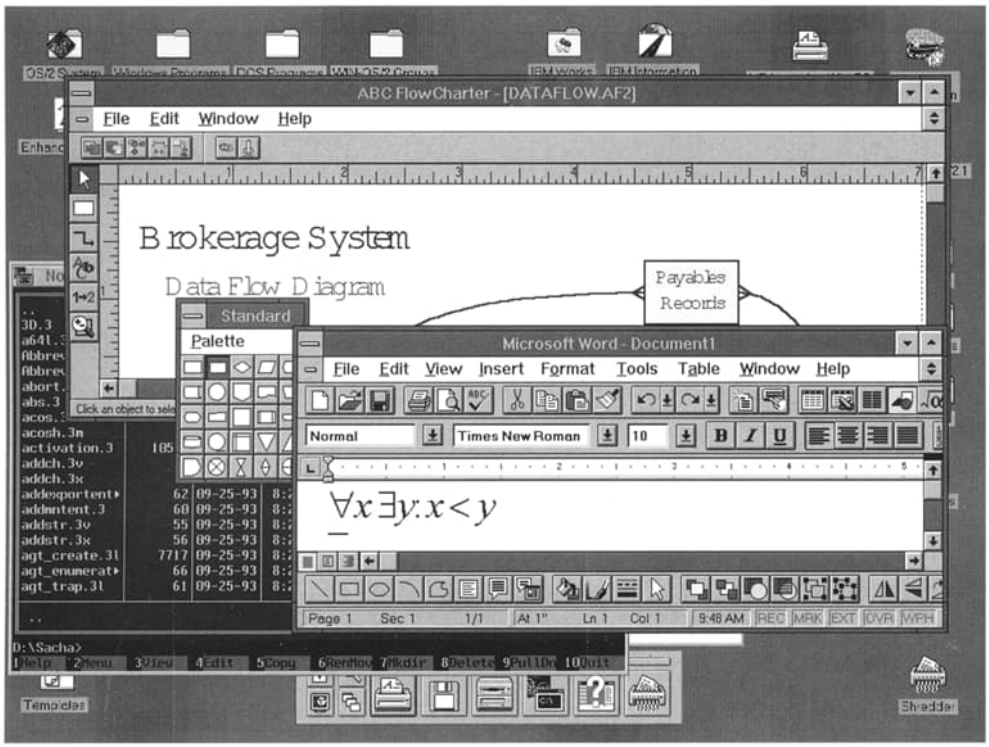


Рис. 1

ное создание, вызов и удаление через сеть и другие продвинутые свойства архитектуры CORBA. При этом следует подчеркнуть, что SOM встроен в систему, а не является какой-либо дополнительной надстройкой. По оценке Гради Буча, «SOM is GOOD».

С пользовательской точки зрения, реакция на Workplace Shell была и остается противоречивой. По мнению «PC Magazine», Workplace Shell является образцом самосогласованности, в то время как комментатору «New-York Times» в обзоре, посвященном OS/2 Warp, непонятно, зачем было менять интерфейс Windows, поскольку он счел, что Warp — это утилита для Windows. Как бы то ни было, попробуем дать словесное описание некоторых характерных приемов работы с объектами системы. Для пользователя *Workplace Desktop* — это *рабочий стол*, содержащий отдельно лежащие или находящиеся в *папках (folders)* документы или папки. Каждый объект может быть представлен различным образом, например папка — в виде списка, набора пиктограмм и так далее. Весьма любопытно, что самому рабочему столу, содержащему все другие визуально идентифицируемые объекты, соответствует каталог DESKTOP на диске, и наоборот, любой каталог в действительности является папкой. Открыв папку *шаблоны (Templates)*, пользователь может нажать правую кнопку «мыши» на одном из них (например на шаблоне папки) и вытащить его, не отпуская кнопки «мыши», на рабочий стол или сразу же поместить в папку (кнопка отжимается). Если вновь созданная папка помещена в визуальный образ какого-либо каталога, создается подкаталог, если в папку на столе — новая папка. Та же операция Drag-and-Drop используется, например, если вы хотите установить в системе новый локальный или сетевой принтер. Любой объект может быть объявлен пользователем шаблоном для создания новых объектов.

У любого визуального объекта щелчком правой кнопки «мыши» открывается так называемое контекстное (всплывающее) меню, характеризующее операции, которые могут быть выполнены над данным объектом. К примеру, у рабочего стола одной из операций будет Shutdown, у программного проекта есть, в частности, Make и MakeMake, у группы выбранных файлов с программами C++ — Delete и Compile. Одна из общих операций над всеми объектами — просмотр или изменение его *конфигурации (settings)*. При выборе этой операции появляется *блокнот (notebook)* с закладками (рис. 2). Здесь можно указать, какой вид примет объект по умолчанию, операцией Drag-and-Drop можно поместить на специальную страницу любую пиктограмму, что изменит представление этого объекта на рабочем столе, и так далее. Разумеется, объекты разных классов могут иметь дополнительные закладки, например, для папки можно установить фон, на котором изображаются пиктограммы, или изменить сортировку по умолчанию.

Все характеристики объектов, изменяемые в их конфигурации, или такие свойства, как расположение или размер папок и даже расположение размещенных пиктограмм, сохраняются в расширенных атрибутах и никогда не исчезают. Более того, даже обычные каталоги файловой системы сохраняют свои атрибуты, например цвет, шрифт, отсортированность и тому подобное. Все эти изменения составляют постепенно формируемый персональный рабочий стол, причем в OS/2

Warp таких конфигураций может быть несколько и одна из них может быть выбрана при загрузке системы. Важно отметить, что даже если HPFS не установлена, все механизмы Workplace Shell остаются в силе, например можно назвать документ «Мое письмо к Юлии». Двойной щелчок по пиктограмме объекта открывает его в виде, выбранном по умолчанию, например, независимо от расширения в имени файла, электронная таблица откроется в правильном виде, поскольку является объектом в иерархии системы. Для любого приложения в старом смысле слова можно создать один или несколько специальных *программных объектов (Program Object)*. В частности, упоминавшиеся параметры настройки сеансов DOS изменяются в *конфигурации объекта*, а сам объект создается с помощью шаблона *программного объекта* и указания в *конфигурации*, что это программа DOS. Когда курсор «мыши» указывает на программный объект, при двойном щелчке, как нетрудно догадаться, запускается отвечающее ему приложение.

Отметим телеграфным «стилем» некоторые другие способы использования системы. Для любого объекта можно создать отражение исходного объекта, помещенное для удобства в какую-либо папку или непосредственно на рабочий стол. Точно таким же образом (нажатием Ctrl-Shift при Drag-and-Drop) в IBM Works производится внедрение объектов в другие. При осуществлении операции Drag-and-Drop перемещаемый и целевой объекты с поразительной скоростью обмениваются сообщениями, предоставляя пользователю информацию о допустимости или недопустимости операции: если целевой объект не способен служить приемником (например, если это папка, которая может содержать лишь письма, а перемещаемый объект — электронная таблица), появляется запретительный знак, если все нормально, вокруг приемника очерчивается прямоугольник, так называемый target emphasis. При изменении имени файла или даже перемещении на другой диск программы, на которую ссылаются *программные объекты*, все ссылки остаются в силе и соответствующим образом отслеживают новое положение и имя исполняемого модуля. Суммируя, можно сказать, что в способах использования OS/2 Workplace Shell наблюдается комбинаторный взрыв в хорошем смысле слова, поскольку реальная жизнь устроена подобным образом, и никому не придет в голову, что надо учиться писать в телефонной книге и в тетради по-разному, хотя это разные объекты, — важно лишь научиться писать. Количество применяемых методов весьма ограничено, однако сочетания методов (регулируемые общностью интерфейсов, а не знанием объектов друг о друге) могут быть самыми непредсказуемыми. Гарантией осмысленности всех операций является согласованность и продуманность интерфейсов — способы и последовательность обращений находятся в руках конечного пользователя.

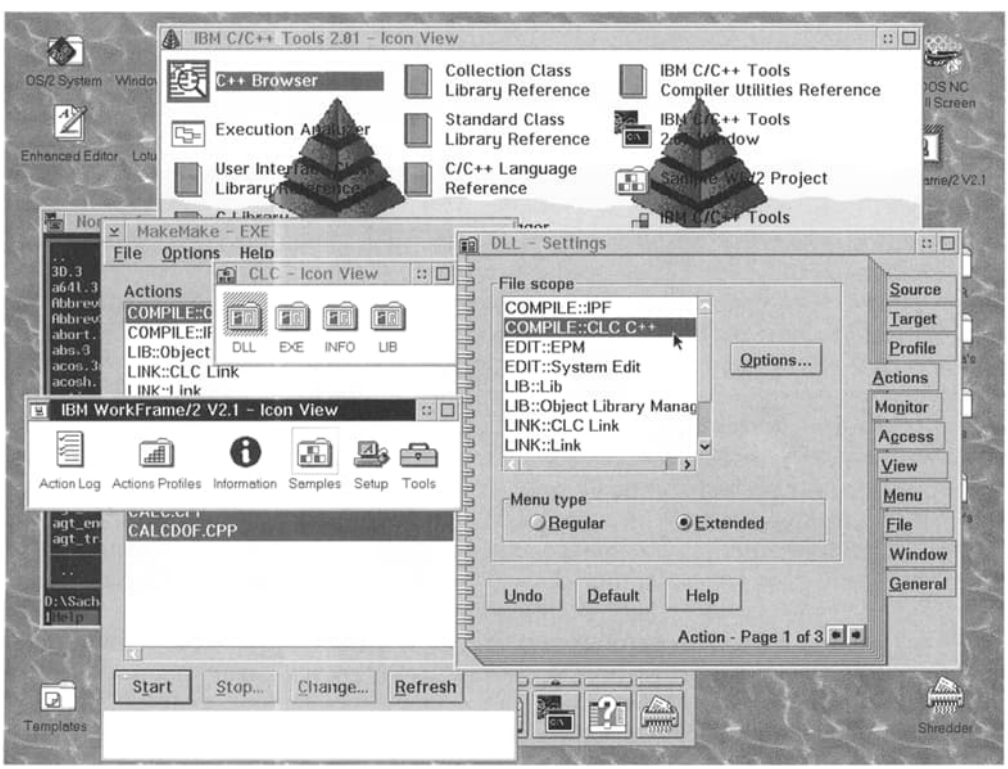


Рис. 2

Новизна OS/2 Warp

По моему мнению, главная особенность OS/2 Warp состоит в том, что существенные нововведения, внесенные в архитектуру системы в предыдущих версиях наконец-то начинают давать заметный выигрыш в производительности и реальной повседневной работе.

– Существенно возросла скорость главных операций,

связанных с заложенными в интерфейс принципами. Например, копирование файлов или объектов на дискету методом Drag-and-Drop в предыдущих версиях занимало много времени, поэтому все использовали Norton Commander для OS/2. Сейчас скорости почти одинаковы. Если раньше открытие папки шаблонов занимало много времени, поскольку за этой операцией стоит много скрытых действий, теперь это происходит почти мгновенно. По имеющейся информации, система работает на машине с оперативной памятью 4 Мб, что заслуживает доверия, поскольку 8 Мб вполне хватало даже для версии 2.11. Тестированная ОС с 16 Мб и полной сетевой поддержкой, включая TCP/IP, проблем с быстродействием не обнаружила.

– В систему введено очень большое количество изменений, улучшающих ее работоспособность с самых разных точек зрения. Выпущенный ранее инструментарий разработки драйверов и увеличение популярности системы позволили улучшить совместимость системы с большим числом SCSI адаптеров, принтеров, видеокарт, устройствами CD-ROM, звуковыми картами и так далее. Не приходится говорить о каких-либо проблемах с режимами HiColor (16 битов) или TrueColor (24 бита) и разрешением на некоторых адаптерах до 1600×1280 пикселов. При установке может быть указан автоматический режим, при котором не только определяется конфигурация системы и внешних устройств и производится оптимизация ОС, но и принимаются разумные предположения о составе устанавливаемых компонентов в зависимости от конфигурации компьютера. При загрузке системы нажатием специальной комбинации клавиш можно вызвать меню, позволяющее не вводить графическую оболочку, загрузить заранее заархивированную копию рабочего стола, вернуться к стандартному общему знаменателю — драйверу VGA и даже включить режим трассировки системного файла CONFIG.SYS для локализации драйвера, вызывающего проблемы при загрузке. Платы PCMCIA можно заменять на ходу, причем с этим событием могут быть ассоциированы определенные автоматические действия, например отправка факсимильных сообщений. Значительно улучшен внешний вид пиктограмм и подбор палитры, так что система внешне напоминает платформу Macintosh. Для быстрого запуска часто используемых программ можно применять LaunchPad. Размещаемые в LaunchPad объекты являются отражениями, и, таким образом, не обязательно служат программными ссылками; будучи объектами системы, они поддерживают вызов контекстных меню, могут участвовать в операциях Drag-and-Drop и так далее. Есть возможности для изменения и создания новых курсоров «мыши», а также использования курсора-кометы, оставляющего след. Пользователи блокнотов оценят новый механизм перемещения объектов — Pickup-and-Drop. Продолжение списка потребовало бы многих страниц.

– С системой OS/2 Warp поставляется так называемый Bonus Pak, содержащий:
  – IBM Works — взаимосвязанную систему шаблонов различных дополнительных объектов, включая базу данных, электронную таблицу, бизнес-графику, календари; пользователь может, например, не открывая ни одного приложения, наложить список рассылки на письмо с полями Mail Merge и затем поместить полученный документ в объект факса; рассылка произведена;
  – пакет доступа к системе Internet предоставляет доступ через модем к одной из служб, включая организованную самой IBM, обеспечивающих доступ к международной сети Internet; в поставку включены все основные утилиты, необходимые для работы в Internet, включая ftp, telnet, gopher, WebExplorer, NewsReader, и тому подобное. Все утилиты, кроме multimedia почты, работают и через непосредственное LAN соединение с Internet (рис. 3); особо надо отметить WebExplorer, хорошо использующий многопоточность и Drag-and-Drop;
  – поддержку рассылки факсов, специализированное приложение для доступа к информационной сети Compuserve и другие.
– Начиная с версии 2.1, в OS/2 включается развитая поддержка multimedia. Предыдущая версия позволяла проигрывать компакт-диски, а также звуковые записи из файлов в различных форматах, включая MIDI, редактировать и смешивать звуковые фрагменты, сопровождать определенные системные операции звуком, проигрывать AVI видеофайлы. OS/2 Warp имеет значительно более мощный multimedia, игровой и образовательный потенциал. Система поддерживает Eastman Kodak Photo CD технологию, что позволяет сохранять и обмениваться со знакомыми семейными и деловыми альбомами фотографий; компонент Person-to-Person дает возможность организовать телеконференцию 8 человек за компьютерами, находящимися в сети, предоставляя в режиме реального времени *совместную рабочую доску (whiteboard)*, на которой можно писать, рисовать, демонстрировать определенные действия с системой, а также с использованием видеокамер проводить

видеоконференции. Кроме того, Warp обеспечивает просмотр большого числа форматов графической, аудио- и видеоинформации, предоставляет драйверы видеооверлейных карт и программное обеспечение для превращения окна системы в телевизор, с помощью системы Video IN позволяет встраивать в презентационное и образовательное обеспечение синхронизированные видеоизображение и звук, поддерживает MPEG, включая просмотр videoCD, и форматы анимаций Autodesk .FLI и .FLC, а также сжатый звуковой формат. Кроме того, в систему встроен высокоскоростной стандарт графических операций — Direct Interface Video Extensions (DIVE), дающий графическим приложениям OS/2 прямой доступ к видеодрайверам, что открывает возможность создания имитационного и игрового программного обеспечения нового поколения. Более того, ожидается появление специализированного инструметария для разработки игр под OS/2 Warp.

Программное обеспечение для OS/2

В противоположность распространенному мнению, гамма программного обеспечения, ориентированного непосредственно на OS/2 и использующего ее возможности, чрезвычайно обширна и сейчас переживает период бурного роста. Из событий осени-94: появились версии для OS/2 Mathematica 2.2, Lantastic Peer-to-Peer Network, VisPro/C, C++ и REXX, IBM Lan Server 4.0, специализированная суперигра «Galactic Civilizations», ожидаются «Doom-2» и «SimCity». Президент фирмы «Corel» в специальном обращении подчеркнул важность OS/2 Warp и объявил о выпуске полного спектра специализированных приложений фирмы к лету 1995 года.

Имеется огромный набор программного обеспечения в областях бизнеса, производства, науки, моделирования. В области свободно распространяемого программного обеспечения перенесены почти все разработки, включая два вида компиляторов C++, Ada, Oberon, два вида свободных 32-битовых Pascal, Perl, Schema. Все известные фирмы, выпускающие C++ компиляторы, исключая «Microsoft», имеют версии для OS/2. «Metaware High C++» оказалась первой, выпустившей так называемый Direct-to-SOM C++ компилятор. IBM C++ компилятор версии 3.0 будет использовать ту же технологию. Имеющиеся в широком спектре CASE-средства бывают как бесплатными, так и очень дорогими. Есть два или три Visual REXX (REXX — это самый мощный встроенный командный язык OS/2). Легко понять, что вопросы, касающиеся прикладных программ, сетевого обеспечения и средств разработки в OS/2, обширны и не могут быть даже вскользь описаны в рамках данной публикации.

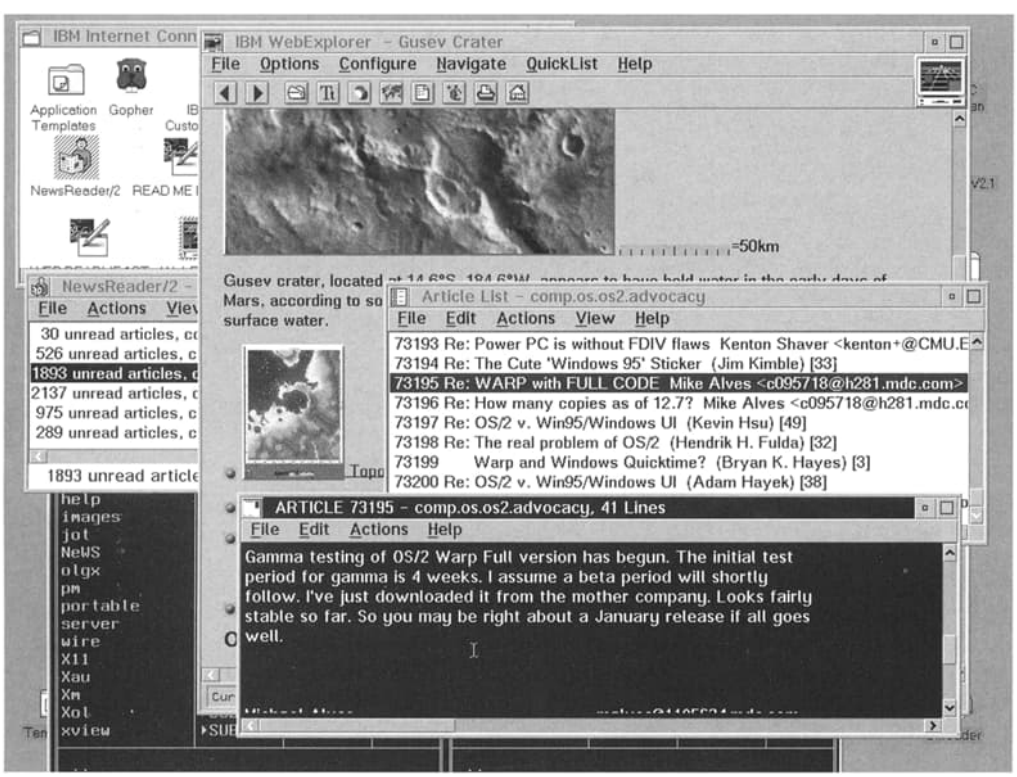


Рис. 3

Автор надеется, что операционная система IBM OS/2 не будет продаваться в России по цене в шесть-восемь раз дороже, чем в США, как было с предыдущими версиями. Кроме того, я выражаю благодарность Николаю Смирнову и Ольге Беликовой из IBM Russia за помощь в подготовке настоящей публикации. ■

INTRALEX
ЮРИДИЧЕСКОЕ ИНФОРМАЦИОННОЕ АГЕНТСТВО

предлагает ПРОГРАММНЫЙ КОМПЛЕКС

ЮСИС

ЮРИДИЧЕСКАЯ СПРАВОЧНО-ИНФОРМАЦИОННАЯ СИСТЕМА

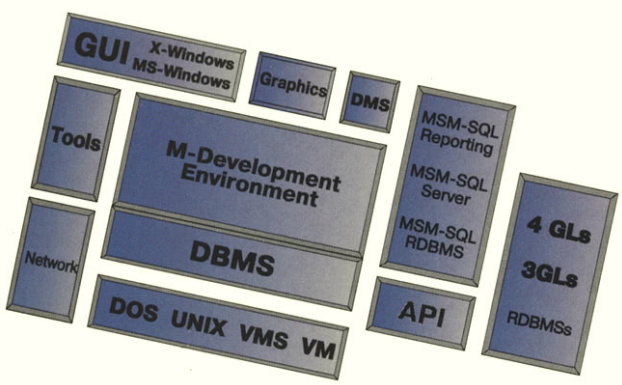
победитель Первого Всероссийского конкурса
правовых баз данных,
первое место в конкурсе "Borland-Contest-94"

ОБНОВЛЕНИЕ БАЗ ДАННЫХ - ЕЖЕДНЕВНО!

Любая дополнительная информация о ПК ЮСИС может быть получена:

в Москве	(095)	288-79-23	в Санкт-Петербурге	(812)	274-81-69
в Архангельске	(818)	43-18-83	в Калининграде	(0112)	43-61-30
в Красноярске	(3912)	23-64-58	в Новосибирске	(3832)	69-36-21
в Ижевске	(3412)	75-86-92	во Владивостоке	(4232)	22-37-55
в Нижнекамске	(84353)	4-09-02	в Калуге	(08422)	3-48-55

Агентство INTRALEX:
Россия, 129110 Москва,
ул. Гиляровского, д.44, офис 57
E-mail: postmaster@intralex@msk.su.



MSM

Работает на любом Hardware:
От I80x86 & Pentium
До Workstation & VAX.
Под любым Software:
От MS-DOS & Windows
До UNIX & VMS.

Наши официальные дистрибьютеры

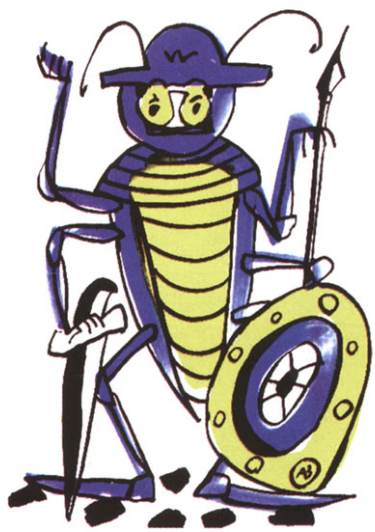
Санкт-Петербург
СП.АРМ
Тел.: (812) 218-7122
Факс: (812) 218-7119
E-Mail:postmaster@
aparm.spb.su

Москва
Escape Ltd
Тел.: (095) 482-1013
Факс: (095) 482-4374
E-Mail:root@
escape.zgrad.su

Воронеж
НПФ "Комета"
Тел.: (0732) 57-1275
Факс: (0732) 57-0406
E-Mail:company@
kometa.voronezh.su

MICRONETICS

Использование возможностей защищенного режима работы процессора в Windows-приложениях



В. Плотников

В статье рассматриваются основные принципы организации защищенного режима работы процессоров Intel. Рассказывается об особенностях функционирования программ среды Windows в защищенном режиме. Статья иллюстрирована задачей форматирования дорожки дискеты, использующей компоненты ядра Windows при программном обслуживании аппаратных средств компьютера в защищенном режиме.

Для того чтобы приступить к обсуждению темы статьи, предлагаю напомнить отдельные положения, связанные с защищенным режимом работы процессора. Это связано с тем, что большинство отечественных программистов предпочитают «старушку-милашку» DOS, в среде которой они обитают, как рыбы в воде. Но хорошая рыба всегда для нагула выходит в «море-окиян». Так давайте попробуем окунуться в тот безграничный океан возможностей, который возник благодаря появлению компьютеров, оснащенных микропроцессорами Intel 80286 или более старшими моделями.

Основные сведения о защищенном режиме работы процессора

Защищенный режим представляет собой организацию вычислительного процесса с целью выполнения независимых программных единиц (задач), использующих возможности, основанные на принципе разделения адресного пространства компьютера и времени. Данный режим характеризуется, в отличие от реального, появлением новых методов организации работы с адресным пространством компьютера и соответственно новых команд процессора, поддерживающих эту вычислительную платформу. Одним из его преимуществ является возможность использования программами всей памяти IBM PC, в том числе и лежащей за границей 1 Мб, без применения различных драйверов, что было характерно для MS-DOS. Работа с памятью может осуществляться путем предоставления ее участков в виде сегментов или страниц (организация последнего распределения подробно описана в [1], [2]).

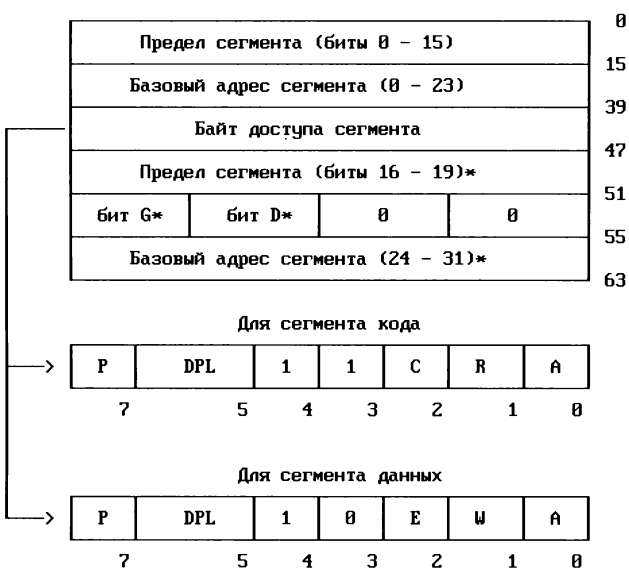
Компьютер после загрузки начинает функционировать в реальном режиме. Для его перевода в защищенный режим любая программа должна открыть адресную линию A20 и выполнить следующую группу команд:

```
mov  ax,1
lmsw ax
```

В реальном режиме для обращения к памяти из программ требовалось загрузить в сегментный регистр номер параграфа начала используемого участка. В защищенном варианте работы процессора адресация памяти осуществляется на основе загрузки в сегментные регистры значений селекторов, которые определяют положение в таблице дескрипторов информации о конкретном участке оперативной памяти, используемом программой. Элемент таблицы дескрипторов или дескриптор сегмента занимает 8 байтов и имеет формат, представленный на рисунке 1. В поле предела сегмента программа должна поместить уменьшенную на единицу длину сегмента в байтах (страницах). Базовый адрес в рассматриваемом случае (сегментной, а не страничной организации памяти) представляет собой физиче-

ский или линейный адрес начала сегмента. Он может быть сформирован в программе на основе логических координат сегмента (номера параграфа и величины смещения) путем сдвига влево на четыре позиции значения параграфа и прибавления к нему смещения. Бит G (гранулярности) определяет, в чем измеряется размер сегмента: если он взведен — в страницах, в противном случае — в байтах. Бит D определяет разрядность команд и данных (если бит взведен, то используется 32-битовая арифметика). Рассмотрим теперь назначение битов поля доступа. Бит P определяет наличие сегмента в памяти, если установлен в 1. Бит обращения к сегменту A устанавливается или сбрасывается автоматически операционной системой. Группа DPL характеризует уровень привилегированности сегмента. Бит C называется битом подчиненности и служит для организации межсегментной передачи управления (подробно об этом см. [3]). Бит R показывает, разрешено ли чтение сегмента кода (R=1) или запрещено. Он может быть использован для организации защиты программ от несанкционированного исследования (модификации) исполняемых кодов задачи. Если сегмент предназначен для хранения данных программы, то бит E должен быть сброшен, а у сегментов стека — взведен. При значении бита W, равном 1, разрешено читать и писать данные в сегмент. При обнуленном бите сегмент должен только читаться. Кроме того, необходимо знать, что запись в сегмент кода запрещена.

Вернемся теперь к подробному рассмотрению представления селектора сегмента. Он имеет разрядность 16 битов, так как загружается в сегментный регистр. Младшие его два бита определяют уровень привилегий (зашиты), запрашиваемых программным объектом. Нулевой уровень высший, а третий низший. Третий бит селектора определяет, в какой таблице дескрипторов содержатся данные по конкретному участку памяти. Если бит взведен, то используется локальная таблица (LDT — Local Descriptor Table), а если сброшен — глобальная таблица (GDT — Global Descriptor Table). В мультизадачной системе может существовать одна таблица GDT, которая находится в ее ядре. Каждая задача может иметь свою локальную таблицу. Старшие 13 битов селектора определяют номер элемента в таблице дескрипторов, характеризующий определенный участок памяти.



* указывает на то, что данные поля дескриптора не используются в защищенном режиме процессором Intel 80286.

Рис. 1. Структура дескриптора сегмента

Защищенный режим не накладывает никаких ограничений на работу с портами компьютера. Можно беспрепятственно использовать команды int и out для доступа к устройствам ПЭВМ. С прерываниями здесь все обстоит по-другому. Они подразделяются на обычные и исключения. Обычные — это реакции системы на внешние события и события конкретной программы (допустим, непосредственная выдача команды int). Исключения возникают при ошибках в момент выполнения программы, связанных с нарушением привилегий защиты. В защищенном режиме существует таблица прерываний, только здесь она является таблицей дескрипторов. Первый 31 элемент зарезервирован для обработки исключительных ситуаций. Остальные свободны и могут быть использованы разработчиками операционных систем или прикладных программ защищенного режима.

Невольно хочется напомнить читателю, что если он пожелает написать простейшую программу, допустим, решающую задачи ввода символьной информации с клавиатуры с ее отображением на терминале, то надо будет написать еще «кусочек» операционной системы, поддерживающий эти действия в защищенном режиме. Поэтому, чтобы не «изобретать велосипед», надо воспользоваться возможностями, которые предоставляют операционные системы OS/2, Windows, UNIX. Для тех, кто желает все-таки сделать свой лицензионно чистый программный продукт, настоятельно рекомендую познакомиться с материалом, изложенным в издании [3]. Кроме того, там можно найти более подробное описание защищенного режима работы процессора, а также познакомиться с приемами и методами создания мультизадачной среды.

Некоторые особенности работы системы Windows в защищенном режиме

Программная среда Windows 3.X является системой, функционирующей в защищенном режиме процессора Intel. Она позволяет организовать

мультизадачность, непосредственную адресацию в приложениях любых участков памяти, в том числе и расширенной. Для задач MS-DOS здесь организуется виртуальная машина, причем Windows эмулирует все прерывания этой ОС. Ядро многооконной среды работает при уровне привилегий 1, а прикладным программам предоставляется только уровень 3. В Windows 3.X в настоящее время используется одна локальная таблица дескрипторов, общая для модулей самой системы и прикладных программ пользователей. Работа с устройствами компьютера здесь организована через BIOS.

Разработчик прикладного программного обеспечения должен отказаться от типичного для MS-DOS приема размещения данных в сегменте кода при создании программ (процедур) для Windows на языке Assembler. Для обращения к участкам памяти компьютера вне области данных приложения (допустим, BIOS, видеообласть) необходимо добавить к локальной таблице дескрипторы, описывающие эти участки памяти, и селекторы дескрипторов загружать в сегментные регистры при обмене информацией с данными участками. Здесь необходимо также заметить, что при функционировании среды Windows в защищенном режиме FAR–указатель на объект программы (процедуру или данные) представляет собой комбинацию селектора и смещения в сегменте. Поэтому для организации DMA-доступа, когда требуется знать физический адрес области памяти, надо по селектору войти в таблицу дескрипторов и оттуда получить базовое значение адреса сегмента.

ПРОГРАММЫ

Windows — это не только прекрасный графический интерфейс, но и вся мощь DOS, базирующаяся на безграничных возможностях организации вычислительного процесса в защищенном режиме.

При программировании в Windows не требуется «вручную» создавать дескрипторы, получать динамические блоки памяти, вызывать прерывания DOS и BIOS. Часть этого реализована в виде функций (процедур) данной среды. Кроме того, дополнительно разработчику предоставляется возможность использования интерфейса DPMI (DOS Protected Mode Interface), предназначенного для поддержки защищенного режима работы процессора как Windows приложений, так и DOS-программ, функционирующих под Windows. Модули DPMI находятся в файлах DOSX.EXE (стандартный режим работы среды) и WIN386.EXE (расширенный режим работы). Обращение к этому интерфейсу происходит посредством вызова прерывания с номером 31h. Функции DPMI позволяют:

- проводить работу с дескрипторами локальной таблицы;
- получать и освобождать блоки динамической памяти;
- обращаться к прерываниям BIOS, DOS, защищенного режима.

Более подробную информацию по функционированию системы Windows 3.X в защищенном режиме можно получить в [4], а познакомиться с функциями DPMI — в [3].

О приемах работы приложений Windows с аппаратными средствами компьютера

Система Windows — мощная операционная среда, дающая возможность пользователю создавать прикладные программы с современным графическим интерфейсом. Эти программы могут обмениваться данными между собой, как и в любой мультизадачной операционной системе, обращаться к последовательному и параллельному портам компьютера, работать с дисковыми файлами. К сожалению, здесь не предусмотрено документированных функций для работы с аппаратными средствами компьютера на низком уровне (программирование контроллеров устройств). Поэтому рассмотрим отдельные приемы, позволяющие из приложения Windows управлять работой устройства на физическом уровне. За основу можно взять процесс создания программы для форматирования дорожки дискеты объемом 1,44 Мб.

Программа форматирования дорожки представляет собой обычное Windows-приложение, к которому предъявляются следующие требования:

- управление работой программы осуществлять посредством выбора пунктов меню;
- определять в диалоге номер форматируемой дорожки, сторону дискеты и символ—заполнитель формата;
- форматирование проводить на основе использования команд контроллера накопителей на гибких магнитных дисках (Floppy Disk Controller — FDC).

На экране должна высвечиваться пиктограмма нашей программы. После ее выбора нажатием клавиши «мыши» появляется меню, содержащие следующие пункты:

1. Перемещение пиктограммы.
2. Пуск программы.
3. О программе.
4. Выход.

Это меню будет формироваться из обычного системного меню основного окна посредством его модификации с помощью функций DeleteMenu, InsertMenu, ModifyMenu как реакция на сообщение WM_CREATE. При выборе пункта «Выход» на экране терминала появляется диалоговое окно (рис. 2). Пользователь может не выгру-

жать из памяти программу, нажав кнопку «Отмена». Пункт «О программе» дает возможность просмотреть описание назначения программы (рис. 3). С помощью пункта 1 пиктограмму задачи можно поместить в любое место окна, в котором была запущена программа. При вызове пункта «Пуск программы» появляется окно сообщения (рис. 4). После нажатия кнопки в нем на экран терминала выводится окно ввода исходных данных (рис. 5). Пользователь должен задать все данные, необходимые для форматирования дорожки, и выбрать «мышью» кнопку «OK». В случае указания номера дорожки, отсутствующего на дискете, или ввода символа-заполнителя не в шестнадцатеричной форме представления на окно ввода накладывается окно с сообщением об ошибке. После корректного задания данных начинается операция форматирования дорожки, по окончании которой появляется окно с сообщением о результатах ее выполнения.

Задача форматирования дорожки реализована в виде следующих процедур языков C и Assembler:

- главной (основной) с именем WinMain (ее текст приведен в листинге 1);
- основного окна приложения WndProc (листинг 1);
- запуска операций по форматированию дискеты (листинг 2);
- окна ввода исходных данных форматирования (листинг 2);
- форматирования дорожки дискеты (листинг 3);
- вспомогательных функций, реализующих программирование FDC (листинг 4).

Процедура WinMain является типичным примером основной программы приложения Windows, создающей основное окно и организующей обработку очереди сообщений.

Функция основного окна WndProc служит для обработки сообщений, поступающих от операционной среды. Так, при создании окна (сообщение WM_CREATE) происходит модификация системного меню задачи. Раскрытие основного окна перехватывается с помощью сообщений WM_SHOWWINDOW и WM_QUERYOPEN. Процедура реагирует на них так, что основное окно остается всегда в виде пиктограммы. Выбор пунктов системного меню пользователем порождает сообщение WM_SYSCOMMAND. Программно анализируется идентификатор пункта (переменная wParam) и происходит высвечивание информационного окна программы, или вызов процедуры запуска операций форматирования, или программа завершает свою работу и выгружает себя из памяти.

Процедура Format_Disk предназначена для создания окна ввода исходных данных для форматирования. В этом окне отображаются кнопки выбора стороны дискеты и подтверждения ввода данных, однострочные окна редактирования (поля

Рис. 2. Диалоговое окно завершения программы

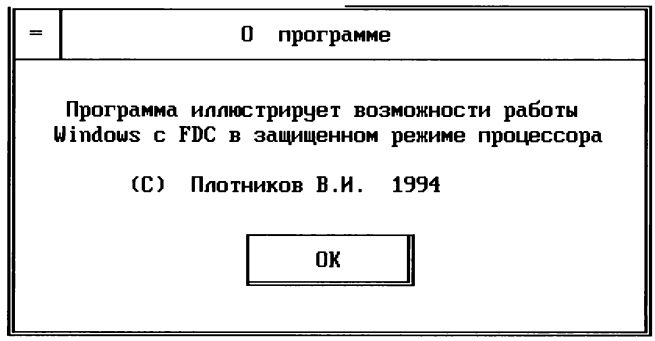


Рис. 3. Окно описания назначения программы

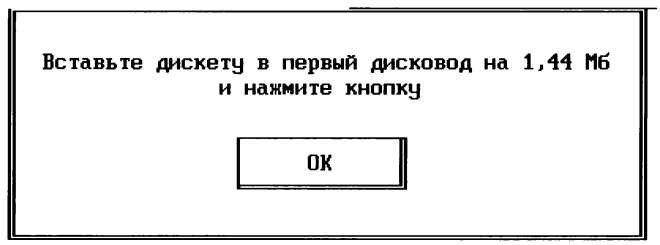


Рис. 4. Окно сообщения пользователю

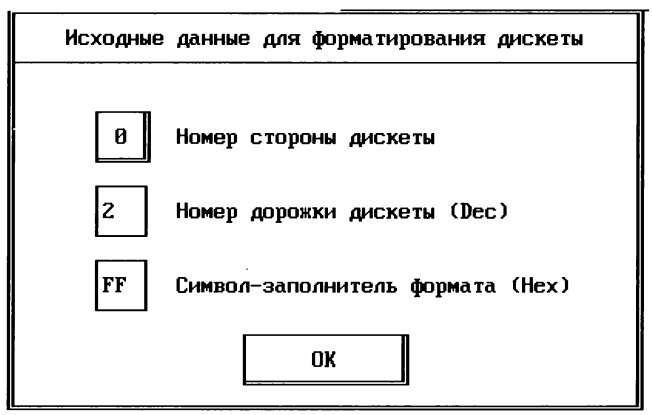


Рис. 5.Окно ввода исходных данных форматирования

ввода) номера дорожки и значения символа-заполнителя при операции форматирования.

Сообщения, поступающие в окно ввода данных, обслуживаются с помощью функции окна WndProc_1. Так при создании окна (сообщение WM_CREATE) осуществляется «выключение» пунктов меню основного окна с целью исключения возможности запуска еще одной процедуры

форматирования. При нажатии на кнопку выбора стороны дискеты (параметр wParam равен ID_BT_1) происходит высвечивание на кнопке номера противоположной стороны магнитного носителя. После выбора «мышью» кнопки подтверждения ввода исходных данных (параметр wParam принимает значение ID_BT_1) осуществляется контроль правильности выбора дорожки дискеты, корректности значения символа-заполнителя. Если пользователь неверно указал эти параметры, то отображается окно сообщений об ошибке. После получения программой исходных данных выделяется динамическая память для размещения буфера DMA с помощью функции GlobalDosAlloc (если для организации прямого доступа к памяти требуется использовать буфер большого размера, рекомендуется получить селектор выделенного блока посредством вызова функции GlobalLock и через DPMI для данного селектора определить физический адрес буфера). Затем управление передается процедуре форматирования дорожки дискеты FormatTrk. После ее завершения анализируется код возврата, освобождается блок динамической памяти и «включаются» пункты меню.

Процедура FormatTrk предназначена для форматирования конкретной дорожки дискеты. Она выполнена на языке ассемблера. В начале своей работы она проверяет наличие защищенного режима процессора. Добавляет к таблице LDT системы дескриптор, описывающий часть области BIOS. Это осуществляется обращением к интерфейсу DPMI. Затем запускается операция форматирования дискеты. После окончания работы процедура передает код возврата функции окна WndProc_1.

Библиотека вспомогательных процедур представляет собой набор модулей Assembler:

- определения фактического адреса буфера DMA-обслуживания (memor);
- определения относительного номера накопителя ПЭВМ размером 3,5 дюйма (opr_nak);
- организации задержки выполнения программы (sader);
- включения мотора дисковода заданного накопителя (motor);
- записи байта информации в регистр данных FDC (fdc_out);
- чтения байта информации из регистра данных FDC (fdc_in);
- организации задержки выполнения программы до момента возникновения прерывания от FDC после завершения дисковых операций (int_wait);
- инициализации внутренних регистров FDC и установки головок дисковода в начальное положение (rekal);
- подвода головок дисковода к выбранной дорожке дискеты (ustan);
- вывода данных в контроллер через второй канал DMA (dma_out);
- форматирования дорожки дискеты (form);
- выключения мотора дисковода заданного накопителя(end_motor).

Процедуры данной библиотеки являются незначительной модификацией аналогичных модулей, рассмотренных в статье [5]. Так, модули opr_nak, motor, end_motor, fdc_out, fdc_in, dma_out не претерпели никаких изменений, поэтому их тексты не приведены в листинге 4. С помощью модуля memor вычисляется фактический номер сегмента начала буфера DMA, так чтобы блок памяти, заказанный для прямого доступа, не пересекал границу сегмента. Для организации временной задержки для приложения Windows пришлось отказаться от «трюка» с командой HLT (см. подобную подпрограмму в [5]): в защищенном режиме эта команда является привилегированной (уровень 0), а наше приложение выполняется на третьем уровне. Модули int_wait, rekal, ustan являются незначительной модификацией аналогичных из [5]. Их изменения также связаны с особенностями защищенного режима: для обращения к области BIOS сегментный регистр должен содержать номер селектора, а не номер параграфа памяти.

Для решения этой задачи надо совместно откомпилировать файлы исходных модулей и файл определения модуля программы (листинг 5). Это наиболее просто осуществить с помощью интегрированной среды Borland C++ 3.1.

В статье были рассмотрены отдельные особенности, которые накладывает защищенный режим на приложения Windows. Данный режим позволяет программам выйти на более высокий уровень. Кроме прекрасного графического интерфейса пользователя, мы получаем и всю мощь DOS через DPMI. Так что многие наработки отечественных программистов, в том числе и по созданию ключевых дискет, основанные на непосредственном программировании контроллера FDC, получат здесь дальнейшее развитие и совершенствование. ■

Литература

1. **Петрухин В.С., Степченков Ю.А., Филин А.В. Персональные ЭВМ на основе архитектуры Intel 80386: [В 2 т.].– Обнинск: ИНВЕСКО, 1993.**
2. **Григорьев В.Л. Микропроцессор i486. Архитектура и программирование : [В 4 т.]/ Т. 1. Программная архитектура.– М.: ГРАНАЛ, 1993.**
3. **Фролов А.В., Фролов Г.В. Защищенный режим процессоров Intel 80286, 80386, 80486. Практическое руководство по использованию защищенного режима.– М.: ДИАЛОГ–МИФИ, 1993.**
4. **Нортон П., Йао П. Программирование на Borland C++ в среде Windows: [В 2 т.]. – Киев: Диалектика, 1993.**
5. **Плотников В.И. Создание ключевых дискет// Монитор.– 1994. – № 5.**

Листинг 1.
Тексты процедур WinMain и WndProc

```
#include <windows.h>
// Идентификаторы пунктов меню
#define  ID_PUSK     101 // Пуск программы
#define  ID_ABOUT    102 // О программе
#define  ID_EXIT     103 // Выход
long FAR PASCAL WndProc(HWND, unsigned, WORD, LONG);
extern  HANDLE hInstance_;
extern  char   cText[200], cWinText[50];
extern WORD wCountItMn; // Число пунктов меню
extern HMENU   hMenu;
HANDLE  hInst;  // Дескриптор текущей копии
int     PASCAL  WinMain
(HANDLE hInstance, HANDLE hPrevinstance,
 LPSTR lpCmdLine, int nCmdShow)
////////////////////////////////////////////////////
//
//       Главная процедура запускаемой задачи
//
////////////////////////////////////////////////////
{
HWND    hWnd;
MSG     message;
WNDCLASS       WndClass;
    hInstance_=hInstance;
// Проверка на наличие копии в памяти
      if (!hPrevinstance)
                              {
//            Создание основного окна
     WndClass.lpszClassName="FDC";
     WndClass.hInstance=hInstance;
     WndClass.lpfnWndProc=(FARPROC) WndProc;
     WndClass.style=NULL;
     WndClass.hbrBackground=NULL;
     WndClass.hCursor=NULL;
     WndClass.hIcon=LoadIcon(NULL, IDI_APPLICATION);
     WndClass.lpszMenuName=NULL;
     WndClass.cbClsExtra=NULL;
     WndClass.cbWndExtra=NULL;
     RegisterClass(&WndClass);
  OemToAnsi
  ("Пример работы с FDC в среде Windows", cText);
 hWnd=CreateWindow
      ("FDC", cText, WS_OVERLAPPED | WS_SYSMENU,
       0, 0, 400, 200, NULL, NULL, hInstance, NULL);
ShowWindow(hWnd, SW_SHOWMINIMIZED); UpdateWindow(hWnd);
// Обработка сообщений
while (GetMessage(&message,NULL,0,0))
      {
      TranslateMessage(&message);
      DispatchMessage(&message);
      }
            return message.wParam;
                              }
      return  NULL;
}
long FAR PASCAL WndProc
(HWND hWnd, unsigned message, WORD wParam, long lParam)
////////////////////////////////////////////////////
//         Функция основного окна задачи
//
////////////////////////////////////////////////////
                              {
void NEAR Format_Disk(HWND);
WORD    wId;
int     iCode;
      switch (message)
{
// Запрещение раскрытия основного окна программы
//       (всегда будет пиктограмма)
            case WM_SHOWWINDOW:
  if(wParam != 0) ShowWindow(hWnd, SW_SHOWMINIMIZED);
            return NULL;
            case WM_QUERYOPEN:
            return NULL;
//    Коррекция пунктов системного меню
            case WM_CREATE:
      hMenu=GetSystemMenu(hWnd, FALSE);
      OemToAnsi("Пуск программы ...", cText);
      while(DeleteMenu(hMenu, 2, MF_BYPOSITION));
      DeleteMenu(hMenu, 0, MF_BYPOSITION);
      InsertMenu(hMenu, -1, MF_BYPOSITION |
      MF_ENABLED | MF_STRING, ID_PUSK, cText);
      OemToAnsi("О программе ...", cText);
      InsertMenu(hMenu, -1, MF_ENABLED | MF_STRING,
      ID_ABOUT, cText);
      OemToAnsi("Выход ...", cText);
      InsertMenu(hMenu, -1, MF_ENABLED | MF_STRING,
      ID_EXIT, cText);
      OemToAnsi("Перемещение пиктограммы",cText);
      wId=GetMenuItemID(hMenu, 0);
      ModifyMenu(hMenu, 0, MF_BYPOSITION |
      MF_ENABLED | MF_STRING, wId, cText);
      wCountItMn=GetMenuItemCount(hMenu);
      return NULL;
//        Обработка выбора пунктов меню
            case WM_SYSCOMMAND:
if(wParam == ID_PUSK){Format_Disk(hWnd); return NULL;}
   if(wParam == ID_ABOUT) {
 OemToAnsi("О программе", cWinText);
 OemToAnsi(
 "   Программа иллюстрирует возможности работы\n"
 "Windows с FDC в защищенном режиме процессора\n\n"
 "                (C)  Плотников В.И.  1994",
        cText);
      MessageBox(hWnd, cText, cWinText, MB_OK);
                        }
     if(wParam == ID_EXIT) {
 OemToAnsi(
 "Прекратить выполнение данной программы", cText);
 iCode=MessageBox(hWnd, cText, " ",
    MB_OKCANCEL | MB_DEFBUTTON1 | MB_SYSTEMMODAL);
         if (iCode == IDOK)  PostQuitMessage(0);
                        }
            default:
 return(DefWindowProc(hWnd, message, wParam, lParam));
}
                        }
```

Листинг 2.
Тексты процедур Format_Disk и WndProc_1

```
#include <ctype.h>
#include <stdlib.h>
#include <windows.h>
#define  ID_ED_1    201
#define  ID_ED_2    202
#define  ID_BT_1    11
#define  ID_BT_2    12
/////////////  Коды возврата  ////////////////////////
#define ERR_01  1    // На машине отсутствует
                     // накопитель на 1,44 Мб
#define ERR_02  2    // Отсутствует дискета в
                     // дисководе
#define NO_ERR  0    // Норма
#define ERR_1   -1   // Задача находится не в защищенном
                      // режиме работы процессора
```

```
   #define ERR_2   -2   // Ошибка при работе в защищенном
                        // режиме
#define ERR_3   -3   // Неисправность контроллера
                     // дисковода
#define ERR_4   -4   // Ошибка дисковой операции
#define ERR_5   -5   // Ошибка при операции
                     //форматирования дорожки
////////////////////////////////////////////////////
#define MEMOR   512  // Длина блока для DMA
HANDLE  hInstance_;
char    cText[200], cWinText[50];
WORD    wCountItMn; // Число пунктов меню
HMENU   hMenu;
HDC             hDC;
int             iHs;
TEXTMETRIC      tm;
LPSTR            cN_D[2]        ={"0","1"};
char            cNmTrk[3]    ="2",
                cChFrmt[3]   ="FF";
BOOL            bN_D    =FALSE;
HWND hWnd_1, hWndEd_1, hWndEd_2, hWndBt_1, hWndBt_2;
long FAR PASCAL WndProc_1(HWND, unsigned, WORD, LONG);
 int NEAR FormatTrk(LPSTR, BOOL, int, BYTE);
      void NEAR Format_Disk(HWND hWnd)
/********************************************** ***
  Процедура запуска операций форматирования дискеты
 ----------------------------------------------------
 Вход:
   hWnd - дескриптор основного окна приложения
**************************************************/
{
WNDCLASS      WndClass;
 OemToAnsi("Вставьте дискету в первый дисковод "
        "на 1,44 Мб и нажмите кнопку", cText);
 MessageBox(hWnd, cText, " ", MB_OK | MB_SYSTEMMODAL);
    hDC=GetDC(hWnd); GetTextMetrics(hDC, &tm);
            ReleaseDC(hWnd, hDC);
  iHs=tm.tmExternalLeading+tm.tmInternalLeading+
     tm.tmHeight;
// Создание окна для ввода исходных данных
  WndClass.lpszClassName="FDC_1";
  WndClass.hInstance=hInstance_;
  WndClass.lpfnWndProc=(FARPROC) WndProc_1;
  WndClass.style=NULL;
  WndClass.hbrBackground=GetStockObject(LTGRAY_BRUSH);
  WndClass.hCursor=LoadCursor(NULL, IDC_ARROW);
  WndClass.hIcon=NULL;
  WndClass.lpszMenuName=NULL;
  WndClass.cbClsExtra=NULL;
  WndClass.cbWndExtra=NULL;
  RegisterClass(&WndClass);
 OemToAnsi(
"Исходные данные для форматирования дискеты", cText);
        hWnd_1=CreateWindow("FDC_1", cText,
        WS_POPUP | WS_THICKFRAME | WS_CAPTION,
         tm.tmMaxCharWidth*8, iHs*6,
         tm.tmMaxCharWidth*28, iHs*12,
        hWnd, NULL, hInstance_, NULL);
         ShowWindow(hWnd_1, SW_SHOWNORMAL);
         UpdateWindow(hWnd_1);
        hDC=GetDC(hWnd_1); GetTextMetrics(hDC, &tm);
         ReleaseDC(hWnd_1, hDC);
  iHs=tm.tmExternalLeading+tm.tmInternalLeading+
     tm.tmHeight;
//  Создание кнопки выбора стороны дорожки
        hWndBt_1=CreateWindow(
         "button", cN_D[bN_D],
        WS_CHILD | BS_DEFPUSHBUTTON | WS_VISIBLE,
        tm.tmMaxCharWidth*4, iHs,
       tm.tmMaxCharWidth*2, iHs+6,
       hWnd_1, ID_BT_1, hInstance_, NULL);
// Создание окна ввода номера дорожки дискеты
       hWndEd_1=CreateWindow(
       "Edit", cNmTrk,
       WS_CHILD | ES_LEFT | WS_BORDER | WS_VISIBLE,
       tm.tmMaxCharWidth*4, iHs*3,
       tm.tmMaxCharWidth*2+1, iHs+4,
       hWnd_1, ID_ED_1, hInstance_, NULL);
// Создание окна ввода значения символа-заполнителя
//                формата
       hWndEd_2=CreateWindow("Edit", cChFrmt,
       WS_CHILD | ES_LEFT | WS_BORDER | WS_VISIBLE,
       tm.tmMaxCharWidth*4, iHs*5,
       tm.tmMaxCharWidth*2+1, iHs+4, hWnd_1,
       ID_ED_2, hInstance_, NULL);
//  Создание окна подтверждения ввода данных
       hWndBt_2=CreateWindow("button", "OK",
       WS_CHILD | BS_DEFPUSHBUTTON | WS_VISIBLE,
       tm.tmMaxCharWidth*12, iHs*8,
       tm.tmMaxCharWidth*4, iHs+6,
       hWnd_1, ID_BT_2, hInstance_, NULL);
       return;
}
long FAR PASCAL WndProc_1 (HWND hWnd,
unsigned message, WORD wParam, long lParam)
////////////////////////////////////////////////////
//
// Функция окна ввода исходных данных форматирования
//
////////////////////////////////////////////////////
                    {          // 1
       int           i, iColor, iCode, iNmTrk, iLn;
       BYTE          byChFrmt;
       WORD           wId;
       DWORD         dwAdrMem;
       char           cBuf[3];
       PAINTSTRUCT   ps;
       switch (message)
{                           //2
            case WM_CREATE:
// Выключение пунктов системного меню программы
   for(i=0; i < wCountItMn; i++)         {
      wId=GetMenuItemID(hMenu, i);
   EnableMenuItem(hMenu, wId,  MF_GRAYED);  }
            break;
            case WM_PAINT:
//         Отображение текста в окне
       hDC=BeginPaint(hWnd, &ps);
       iColor=GetBkMode(hDC);
       SetBkMode(hDC, LTGRAY_BRUSH);
       OemToAnsi("Номер стороны дискеты", cText);
       TextOut (hDC, tm.tmMaxCharWidth*8, iHs*1+3,
       cText, lstrlen(cText));
       OemToAnsi("Номер дорожки (Dec)", cText);
       TextOut (hDC, tm.tmMaxCharWidth*8, iHs*3+3,
       cText, lstrlen(cText));
       OemToAnsi("Символ-заполнитель формата (Hex)",
       cText);
       TextOut (hDC, tm.tmMaxCharWidth*8, iHs*5+3,
       cText, lstrlen(cText));
       SetBkMode(hDC, iColor); EndPaint(hWnd, &ps);
       break;
            case WM_COMMAND:
//-----------------------------------------------------
// Обработка нажатия кнопки выбора номера стороны
//              дискеты
//-----------------------------------------------------
       if (wParam == ID_BT_1) {
  bN_D=!bN_D; SetWindowText(hWndBt_1, cN_D[bN_D]);
            return NULL;
```

```
                    }
//-------------------------------------------------------
// Обработка нажатия кнопки подтверждения выбора
//            исходных данных
//-------------------------------------------------------
   if (wParam == ID_BT_2)
                    {                          // 3
// Анализ введенной пользователем информации
   if ( SendMessage
   (hWndEd_1, WM_GETTEXTLENGTH, 0, 0L)) {     //4
  iLn=SendMessage(hWndEd_2, WM_GETTEXTLENGTH, 0, 0L);
         if ( iLn != 0)          {           // 5
   SendMessage(hWndEd_1, WM_GETTEXT, 3, (LPSTR) cBuf);
              iNmTrk=atoi(cBuf) ;
            lstrcpy(cNmTrk, cBuf);
    if( (lstrcmp("00", cBuf) != 0) &&
       (lstrcmp("0 ", cBuf) != 0) &&
       (lstrcmp(" 0", cBuf) != 0) &&
       (lstrcmp("0", cBuf) != 0)     )
                                  {        //6
                if( iNmTrk == 0 ) {        // 7
      OemToAnsi("Указанный номер дорожки "
  "не является десятичным числом", cText);
      MessageBox(hWnd, cText, NULL, MB_OK);
      return NULL;
                              }            // 7
                                 }         //6
  if( iNmTrk < 0 || iNmTrk > 79) {         // 8
OemToAnsi( "Номер дорожки может изменяться \n"
"                от 0 до 79", cText);
MessageBox(hWnd, cText, NULL, MB_OK);
      return NULL;
                           }               // 8
   if( iLn != 2) {                          // 9
      OemToAnsi(
"Символ-заполнитель формата должен задаваться"
" двумя шестнадцатеричными цифрами", cText);
      MessageBox(hWnd, cText, NULL, MB_OK);
      return NULL;
            }                              // 9
   SendMessage(hWndEd_2, WM_GETTEXT, 3, (LPSTR) cBuf);
          lstrcpy(cChFrmt, cBuf);
     if( (isxdigit(cChFrmt[0]) == 0) ||
        (isxdigit(cChFrmt[1]) == 0) )   {
      OemToAnsi(
"Символ-заполнитель не является представлением"
" шестнадцатеричного числа", cText);
      MessageBox(hWnd, cText, NULL, MB_OK);
      return NULL;
                                  }
            AnsiUpper(cChFrmt);
 if(cChFrmt[0] >= 'A') byChFrmt=10+cChFrmt[0]-'A';
             else byChFrmt=cChFrmt[0]-'0';
      byChFrmt=byChFrmt*16;
if(cChFrmt[1] >= 'A') byChFrmt+=10+cChFrmt[1]-'A';
             else byChFrmt+=cChFrmt[1]-'0';
// Выделение памяти для DMA для работы с дискетой
      dwAdrMem=GlobalDosAlloc(MEMOR*2+5);
      if (dwAdrMem == NULL) {              // 10
     OemToAnsi( "Windows не может выделить память "
     "задаче для работы с FDC", cText);
      MessageBox(hWnd, cText, NULL, MB_OK);
      return NULL;
                        }                  // 10
                        else  {            // 11
// Вызов функции для форматирования дорожки дискеты
 iCode=FormatTrk(dwAdrMem, bN_D, iNmTrk, byChFrmt);
// Разблокирование и освобождение блока памяти
      GlobalDosFree((WORD) dwAdrMem);
// Анализ кода возврата
  if(iCode == NO_ERR)
 {
OemToAnsi("Операция с дискетой выполнена успешно",
cText); MessageBox(hWnd, cText, " ", MB_OK);
 } else
       {                                   // 12
 if(iCode == ERR_01)
 OemToAnsi(
"На машине отсутствует накопитель 1,44 Мб", cText);
 if(iCode == ERR_02)
OemToAnsi( "Отсутствует дискета в дисководе", cText);
 if(iCode == ERR_1)
      OemToAnsi(
"Задача предназначена только для работы в "
"защищенном режиме работы процессора", cText);
 if(iCode == ERR_2)
      OemToAnsi(
"Возникла ошибка при работе с локальными "
"таблицами дескрипторов памяти", cText);
 if(iCode == ERR_3)
OemToAnsi(
"Неисправность контроллера дисковода", cText);
  if(iCode == ERR_4)
      OemToAnsi("Ошибка дисковой операции", cText);
  if(iCode == ERR_5)
 OemToAnsi("Ошибка операции форматирования", cText);
      MessageBox(hWnd, cText, NULL, MB_OK);
                                  } // 12
                           }       // 11
// Включение пунктов системного меню программы
 for(i=0; i < wCountItMn; i++)
 { wId=GetMenuItemID(hMenu, i);
      EnableMenuItem(hMenu, wId, MF_ENABLED);}
// Разрушение окна ввода исходных данных
                 DestroyWindow(hWnd);
                 return NULL;
                              }       // 5
                           }     // 4
      OemToAnsi(
"Заданы не все исходные данные для форматирования"
" дискеты", cText);
      MessageBox(hWnd, cText, NULL, MB_OK);
      return NULL;
                 }                 // 3
            default:
return(DefWindowProc(hWnd, message, wParam, lParam));
} //2
                  } // 1
```

Листинг 3

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
; Процедура форматирования дорожки дискеты  1,44 Мб
;
;=======================================================
;
;    Соглашение о связях:
;
; данная процедура, выполненная на языке Assembler,
; реализована в виде функции языка C.
;
;-------------------------------------------------------
;
;      Вход:
;
;   Первый параметр определяет адрес  выделенного
; блока  памяти  для  обслуживания канала DMA дискеты
; (в старшем слове - сегмент, а в младшем - селектор).
;   Второй параметр задает номер стороны дискеты.
```

```
;   Третий параметр определяет номер форматируемой
; дорожки дискеты.
;  Четвертый параметр определяет символ-заполнитель
;при операции форматирования.
;
;     Выход:
;
;   Функция возвращает величину кода возврата,
;характеризующего результат выполнения операций по
;форматированию дорожки дискеты.
;
;
;-------------------------------------------------
;
;    Объявление данной процедуры в языке C
;         (для Windows-приложения):
;
;    int NEAR FormatTrk(LPSTR, BOOL, int, BYTE)
;
;=================================================
;
;        (c)   Плотников В.И.    1994
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
      .MODEL  SMALL      ; Определяет модель памяти
                         ; вызывающей программы
      .DATA
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
;           Блок переменных процедуры
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
dma_adr dw       0           ; Сегментный адрес буфера DMA
off_dma_adr      dw           0
nom_nak db       0           ; Номер накопителя 1,44 Мб
form_buf         db          4*18 dup(0)
selector         dw          0       ; Селектор области BIOS
buf_deskr        label       byte    ; Дескриптор сегмента
      dw         0100h       ; Предел сегмента
      dw         0400h       ; Базовый адрес
      db         0           ; BIOS
      db         11110010b   ; Поле доступа
      db         0
      db          0
;++++++++++++++++++++++++++++++++++++++++++++++++++++++++
;
;      Описание констант процедуры
;
;++++++++++++++++++++++++++++++++++++++++++++++++++++++++
GAP3   equ    50h
;========      Коды возврата  ======================
ERR_01 equ      1        ; На машине отсутствует
                         ; накопитель 1,44 Мб
ERR_02 equ      2        ; Отсутствует дискета в
                         ; дисководе
NO_ERR equ      0        ; Норма
ERR_1  equ      1         ; Задача находится
                         ; не в защищенном
                         ; режиме работы процессора
ERR_2  equ       2       ; Ошибка при работе
                         ; в защищенном режиме
ERR_3  equ      3        ; Неисправность контроллера
                         ; дисковода
ERR_4  equ       4       ; Ошибка дисковой операции
ERR_5  equ      5        ; Ошибка форматирования
      .CODE
      PUBLIC  _FormatTrk
_FormatTrk     PROC
ARG Adr_Buf:WORD:2,bN_D:WORD,iNmTrk:word,byChFrmt:byte
      cld
      push    bp
      mov     bp,sp
      push    di
      push    si
      push    ds
;
; Получаем физический адрес буфера DMA
;
      mov     cx,4*18
      mov     ax,Adr_Buf+2
      call    memor
      mov     dma_adr,ax
      mov     off_dma_adr,cx
;
; Определяем номер селектора области BIOS
;
      mov     ax,1686h
      int     2Fh     ; Защищенный режим?
      or      ax,ax
      jz      cr_24   ; Да
      mov     ax,ERR_1
      jmp     cr_23
cr_24:
      mov     ax,0
      mov     cx,1
      int     31h     ; Создаем дескриптор
                      ; в таблице LDT
      jnc     cr_22
      mov     ax,ERR_2
      jmp     cr_23
cr_22:
      mov     selector,ax
      mov     bx,ax
      mov     ax,000Ch
      mov     di,ds
      mov     es,di
      mov     di,offset buf_deskr
      int     31h     ; Заполняем дескриптор
                      ; в таблице LDT
      jnc     cr_20
      mov     ax,ERR_2
      jmp     cr_11
;
; Готовим буфер с данными для форматирования
;             дорожки дискеты
;
cr_20:
      mov     si,offset form_buf
      mov     cx,18
      mov     ax,1
      mov     bh,byte ptr bN_D
      mov     bl,byte ptr iNmTrk
cr_2:
      mov     word ptr [si],bx
      mov     byte ptr [si+2],al
      mov     byte ptr [si+3],2
      add     si,4
      inc     ax
      loop    cr_2
;
; Определяем номер накопителя 1,44 Мб
;
      call    opr_nak
      cmp     al,0FFh
      jnz     cr_5
      mov     ax,ERR_01
      jmp     cr_11
cr_5:
      mov     nom_nak,al
cr_4:
;
;      Сброс контроллера дисковода
```

```
        mov     cx,3
cr_6:
        mov     al,nom_nak
        or      al,8
        mov     dx,3F2h
        out     dx,al   ; Запись в регистр управления
        call    sader
;
; Включение контроллера и мотора дисковода
;
        mov     al,nom_nak
        mov     bx,selector
        call    motor

      Рекалибровка дисковода

        mov     al,nom_nak
        mov     bx,selector
        call    rekal
        jnc     cr_7
        loop    cr_6    ; Попытка 3 раза провести
                        ; рекалибровку
        mov     ax,ERR_3
        jmp     cr_3
;
; Установка головок на форматируемую дорожку
;
cr_7:
        mov     al,byte ptr bN_D
        shl     al,2
        or      al,nom_nak
        mov     dl,byte ptr iNmTrk
        call    ustan
        jnc     cr_8
        mov     ax,ERR_4
        jmp     cr_3
;
; Проверка наличия дискеты в дисководе
;
cr_8:
        mov     dx,3F7h
        in      al,dx   ; Чтение регистра состояния
                        ; входа
        test    al,80h
        jz      cr_15   ; Есть дискета в дисководе
        mov     ax,ERR_02
        jmp     cr_3
;
; Форматирование дорожки дискеты
;
cr_15:
      mov   si,offset form_buf
      mov   di,off_dma_adr
      mov   ax,Adr_Buf
      mov   es,ax
      mov   cx,4*18
      rep   movsb
      mov   al,byte ptr bN_D
      shl   al,2
      or    al,nom_nak
      mov   bh,GAP3
      mov   bl,byChFrmt
      mov   dx,dma_adr
      mov   ah,2
      mov   cx,4*18
      mov   di,selector
      call  form
      jnc   cr_12
      mov   ax,ERR_5
      jmp   cr_3
```

```
cr_12:
      mov   ax,NO_ERR
;
;     Завершение работы программы
;
cr_3:
      push  ax
      mov   al,nom_nak
      call  end_motor       ; Выключение мотора
      pop   ax
cr_11:
      push  ax
      mov   ax,0001h
      mov   bx,selector
      int   31h             ; Удаление дескриптора BIOS
                            ; из LDT
      pop   ax
cr_23:
      pop   ds
      pop   si
      pop   di
      pop   bp
      ret
      include bibl_win.inc  ; Подключение процедур
                            ; библиотеки
                            ; вспомогательных
                            ; модулей
_FormatTrk    ENDP
      END
```

■

Листинг 4. Тексты библиотеки вспомогательных процедур

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
; Процедура организации задержки выполнения программы
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
sader   proc    near
        push    cx
        mov     cx,0FFFh
sad_1:
        loop    sad_1
        pop     cx
        ret
sader   endp
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
; Процедура ожидания выполнения прерывания контроллера
;
;-------------------------------------------------------
;
;     Вход:
;
; bx -  номер селектора для доступа к области BIOS
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
int_wait      proc    near
      push    es
      push    ax
      mov     es,bx
intwt_2:
      test    byte ptr es:[3Eh],80h
      jnz     intwt_1
      jmp     intwt_2
intwt_1:
      and     byte ptr es:[3Eh],7Fh
```

```
        pop    ax
        pop    es
      ret
int_wait      endp
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
;      Процедура рекалибровки дисковода
;
;-----------------------------------------------------
;
;      Вход:
;
; al -  номер накопителя,
; bx -  номер селектора для доступа к области BIOS.
;
;      Выход:
;
; CF -  если взведен этот флаг, то операции завершились
;      аварийно
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
rekal proc       near
      push      ax
      push      cx
      push      dx
      mov        cl,al
      push      ax
      push      es
      mov        es,bx
      mov       byte ptr es:[40h],0FFh
      pop        es
      pop       ax

      Установка скорости обмена информацией

      mov       dx,3F7h
      mov        al,0
      out       dx,al
;
; Программирование внутренних таймеров контроллера
;
      mov        al,3
      call      fdc_out
      mov       al,0AFh
      call      fdc_out
      mov       al,4
      call      fdc_out

      Рекалибровка дисковода

    mov      al,7
      call      fdc_out
      mov       al,cl
      call      fdc_out
      call      int_wait
      call       sader
      mov       al,08h
      call      fdc_out ; Получение результата прерывания
      call      fdc_in
      push      ax
      call      fdc_in
      pop       ax
      and       al,0C0h ; Анализ выполнения
                        ; операции
      jz        rekl_1
      mov        al,7
      call      fdc_out
      mov       al,cl
      call      fdc_out
      call    int_wait
      call   sader
      mov    al,08h
      call   fdc_out      ; Получение результата прерывания
      call   fdc_in
      push   ax
      call   fdc_in
      pop    ax
      and    al,0C0h     ; Анализ выполнения
                         ; операции
      jz     rekl_1
      stc
      jmp    rekl_2      ; Выход по ненормальному завершению
rekl_1:
      clc
rekl_2:
      pop    dx
      pop    cx
      pop    ax
      ret
rekal   endp
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
; Процедура установки головок на дорожку определенной
;           стороны накопителя
;
;-----------------------------------------------------
;
;      Вход:
;
; al -  номера стороны дискеты и накопителя в формате
;      команд FDC (младшие три бита, остальные 0),
; dl -  номер дорожки,
; bx -  номер селектора для доступа к области BIOS.
;
;      Выход:
;
; CF -  если взведен этот флаг, то операции завершились
;      аварийно
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
ustan proc   near
        push     ax
        push    cx
        push    dx
        mov     cl,al
        push    ax
        push    es
        mov     es,bx
        mov     byte ptr es:[40h],0FFh
        pop     es
        pop      ax
        mov     al,0Fh
        call    fdc_out      ;  Организация
        mov     al,cl        ; выдачи в FDC
        call    fdc_out      ; команды
        mov     al,dl        ; "Установить
         call    fdc_out     ; головки"
        call    int_wait
        call     sader
        mov     al,08h
        call     fdc_out          ; Получение результата прерывания
         call   fdc_in
        push    ax
        call     fdc_in
        pop     ax
        and     al,0C0h      ; Анализ выполнения
                             ; операции
        jz       ust_2
        stc
      jmp    short ust_1
```

```
ust_2:
      clc
ust_1:
      pop    dx
      pop    cx
      pop    ax
      ret
ustan  endp
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
;     Процедура форматирования дорожки дискеты
;
;---------------------------------------------------------
;
;     Вход:
;
; al -  номера стороны дискеты и накопителя в формате
;       команд FDC (младшие три бита, остальные 0),
; ah - код длины сектора,
; bl -  символ заполнения форматирования,
; bh - величина GPL для форматирования,
; cx - длина в байтах буфера с параметрами форматиро-
;      вания дискеты,
; dx - сегментный адрес буфера с параметрами формати-
;      рования дискеты,
; di - номер селектора для доступа к области BIOS.
;
;     Выход:
;
; CF -  если взведен этот флаг, то операции завершились
;     аварийно (нет дискеты)
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
form   proc   near
       push   bx
       push   cx
       push   dx
       push   ax
       call   dma_out     ; Подготовка DMA для передачи
                          ; блока параметров в FDC
       push   es
       mov    es,di
       mov    byte ptr es:[40h],0FFh
       pop    es
       mov    al,4Dh      ; Код операции форматирования
       call   fdc_out
       pop    ax
       mov    dl,ah
       call   fdc_out     ; Выдача номера головки и
                          ; носителя
       mov    al,dl
       call   fdc_out      ; Указание кода длины сектора
       mov    ax,cx
       shr    ax,2        ; Вычисление числа секторов
                          ; на дорожке
       call   fdc_out
       mov    al,bh
       call   fdc_out     ; Указание GPL форматирования
       mov    al,bl
       call   fdc_out     ; Значение символа-
                          ;заполнителя форматирования
       mov    bx,di
       call   int_wait
       call   fdc_in
       push   ax
       mov    cx,6
frm_1:
       call   fdc_in
       loop   frm_1
       pop    ax
       and    al,11000000b    ; Анализ выполнения
                              ; операции
       jz     frm_2
       stc
       jmp    frm_3
frm_2:
       clc
frm_3:
       pop    dx
       pop    cx
       pop    bx
       ret
form   endp
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
; Процедура определения фактического адреса буфера
;                DMA-обслуживания
;---------------------------------------------------------
;
;     Вход:
;
; ax содержит сегментный адрес выделенного Windows
;   блока памяти,
; cx - длина буфера в байтах.
;
;     Выход:
; ax содержит фактический сегментный адрес
;    буфера DMA,
; cx - смещение буфера DMA в выделенном сегменте.
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
memor proc      near
       push      bx
       push      dx
       mov       bx,ax
       mov       dx,bx
       shl       bx,4
       shr       dx,12
       add        bx,cx
       adc       dx,0          ; В (dx,bx) линейный адрес
                               ; блока памяти
       jnc        mem_1
       mov       bx,ax
       shl       bx,4
       xor        cx,cx
       sub       cx,bx
       mov       ax,dx
       jmp        mem_2
mem_1:
       xor       cx,cx
mem_2:
       pop        dx
       pop       bx
       ret
memor  endp
```

Листинг 5.
Текст файла определения модуля

```
NAME            WIN_FDC
DESCRIPTION      'Пример работы с FDC в среде Windows'
EXETYPE          WINDOWS
CODE            MOVEABLE
DATA             MOVEABLE MULTIPLE
HEAPSIZE        1024
STACKSIZE        8192
EXPORTS
            WndProc       @1
            WndProc_1     @2
```

■

COMTEK

COMTEK '95 *24-28 апреля*

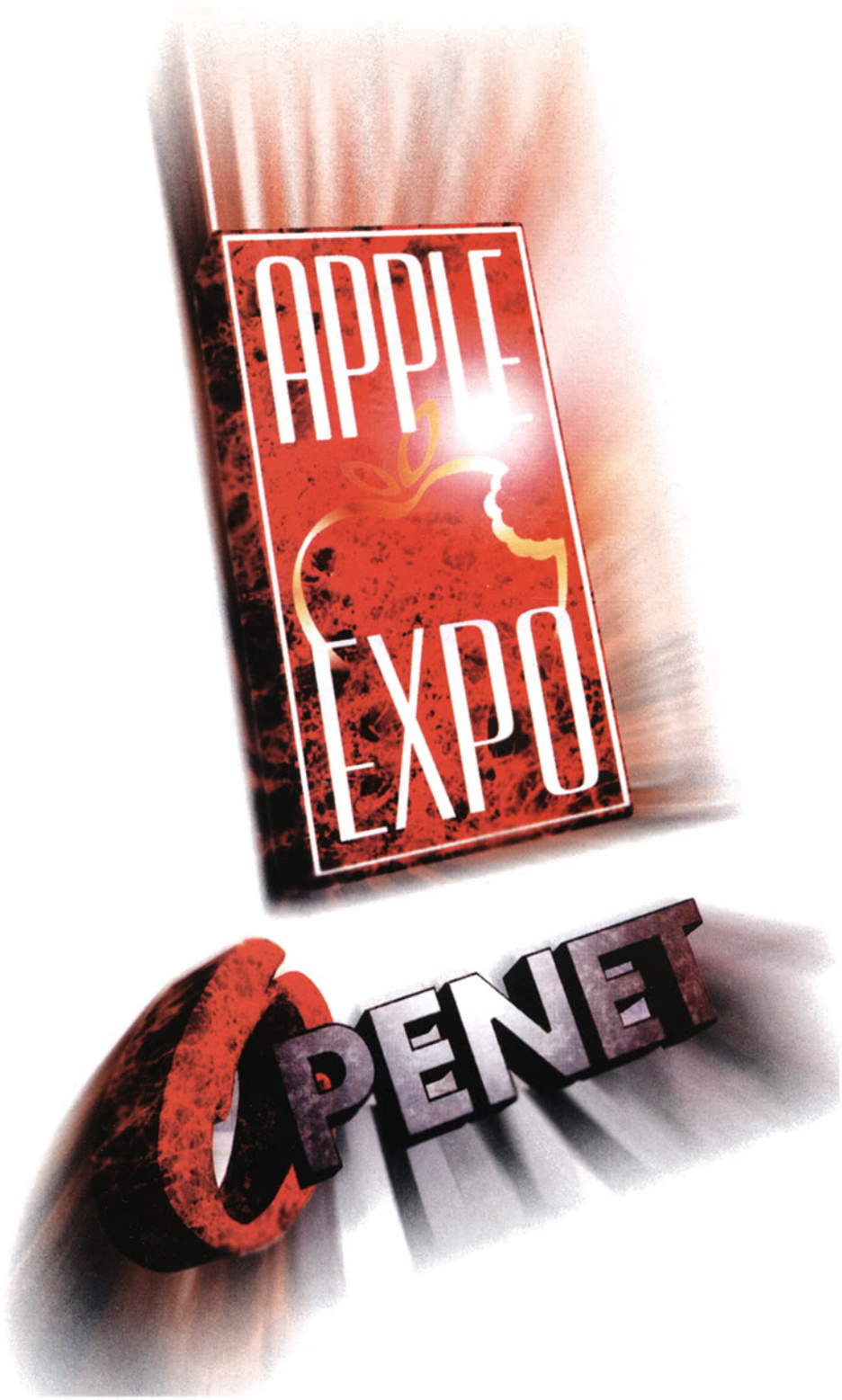
Организаторы выставки:

C·O·M T E K INTERNATIONAL

CROCUS International

6 - Я МЕЖДУНАРОДНАЯ СПЕЦИАЛИЗИРОВАННАЯ ВЫСТАВКА КОМПЬЮТЕРНОЙ ТЕХНИКИ, КОММУНИКАЦИЙ И КОНФЕРЕНЦИЙ.

Выставка расположена в павильонах № 2,3в 1/2павильона"Форум".



APPLE EXPO

OPENET

COMTEK INTERNATIONAL

CROCUS International

На выставке "КОМТЕК '95" развернутся экспозиции **OPENET** и **APPLE EXPO**.

Выставки расположены:
OPENET в павильоне № 1. APPLEEXPO в 1/2 павильона "Форум".

Справки,
дополнительная информация.
Телефон: (095) 249-86-06,
факс: (095) 249-86-09.

Начала программирования в XWindow

Н. Кожекин

Статья содержит рекомендации по программированию в XWindow и использованию разделяемой памяти. Приводятся листинги двух программ, общающихся при параллельной работе. Демонстрируются возможности системы LINUX. Используется компилятор GNU-C++.

ПРОГРАММЫ

Обычно программирование под XWindow однозначно связывают с использованием X Toolkit Intrinsics или OSF/Motif. Однако мне кажется, что лучше начинать с программирования со стандартными библиотеками Xlib и Xutil. Тем более в данном случае это только сэкономит наше время. Дело в том, что на моей машине лишь 4 Мб оперативной памяти, и, следовательно, время работы компилятора для меня критично. Но в любом случае использование Xt не сделает приводимую далее программу более лаконичной.

Однако хватит псевдофилософских рассуждений — лучше для начала разберемся с использованием разделяемой памяти. При этом не стоит забывать о том, что у LINUX ядро пассивно, то есть оно не будет делать ничего самостоятельно, пока мы его не попросим. Для обмена данными между процессами в LINUX реализовано два метода: очереди сообщений и разделяемая память. Мы остановимся на втором из них, так как он является, пожалуй, наиболее мощным. В терминологии UNIX средства межпроцессорного обмена данными называются IPC (InterProcess Communications). Для IPC были разработаны стандартные средства запрещения-разрешения той или иной операции. Они называются семафорами. Для их использования нам нужно подключить к программе следующие файлы определений:

```
#include<sys/types.h>
#include<sys/ipc.h>
#include<sys/sem.h>
```

И тогда мы сможем использовать два системных вызова: **semget** для создания массива семафоров и **semctl** для получения значений семафоров и управления их состоянием. Чтобы узнать о параметрах этих процедур, наберите в системе команду man semget, а чтобы установить комбинацию флагов для доступа, пользуйтесь таблицей 1. Аналогично построена работа с разделяемой памятью. Вы должны подключить файл:

```
#include<sys/shm.h>
```

Можете использовать системные вызовы **shmget** для получения доступа к сегменту, **shmat** для получения адреса, **shmdt**, в сущности, обратный **shmat**, и **shmctl** для изменения режима доступа или удаления сегмента, а флаги доступа полностью аналогичны указанным в таблице 1. При этом для работы со структурой semun я написал две простые, но полезные функции, которые вы можете увидеть в листинге 1.

Теперь можно приступать к программированию собственно под XWindow. Сформулируем цель: написать две программы, каждая из которых открывает свое окно, причем в одном окне можно что-нибудь

Таблица 1

Маски семафоров

SEM_R	00400	владелец читает
SEM_W	00200	владелец пишет
(SEM_R >> 3)	00040	группа читает
(SEM_R >> 3)	00020	группа пишет
(SEM_R >> 6)	00004	остальные читают
(SEM_R >> 6)	00002	остальные пишут
IPC_PRIVATE		личный
IPC_CREATI		создать, если не существует

написать, а в другом это «что-нибудь» появится. Окна в XWindow бывают двух типов: InputOutput и InputOnly. Безусловно, нам нужны первые (последние используются, например, в популярной программе Xeyes). Сделаем в нашей программе еще три включения:

```
#include<X11/Xlib.h>
#include<X11/Xutil.h>
#include<X11/Xos.h>
```

После этого мы можем честно пользоваться всеми благами жизни. Программы под XWindow обычно выглядят так: сначала мы подключаемся к X-серверу (XOpenDisplay), потом получаем номер основного экрана (DefaultScreen), создаем окно (XCreateSimpleWindow), задаем рекомендации для менеджера окон (листинг 2: целесообразно написать процедуру SetWindowManadgerHints, поддерживающую разные версии XWindow, — для LINUX это, в принципе, неактуально, но для общей солидарности UNIX-мира полезно), выбираем события для обработки (XSelectInput), показываем окно (XMapWindow) и приступаем к обработке событий. Для работы с клавиатурой подключим:

```
#include<X11/keysym.h>
```

Именно так в программе-клиенте было реализовано соответствие между клавишами Left Ctrl — Left Shift-A и строкой «EXIT», я имею в виду процедуру XRebindKeySym. Таким образом, при нажатии этой комбинации клавиш в программе-клиенте обе программы благополучно окончат работу. Эти идеи реализованы в листингах 3 и 4.

Одним из основных понятий при работе с графикой в системе XWindow является **графический контекст** (Graphic Context — GC). Он задается процедурой XCreateGC, при этом значения описываются следующей структурой.

```
typedef struct{
int    function;
/* Режим рисования. По умолчанию GXCopy.
Устанавливается процедурой XSetFunction. */
unsigned long    plane_mask;
/* В каких битовых плоскостях идет рисование */
unsigned long    foreground;
unsigned long    background;
/* Цвет плана и цвет фона устанавливаются XSetForeground
и XSetBackground соответственно */
int      line_width;  /* Толщина линии в пикселах */
int     line_style;   /* Тип линии */
int      cap_style;       /* Вид линии на концах */
int     join_style;   /* Вид соединения линий */
int     fill_style;   /* Способ закраски устанавливается*/
                      /* XSetFillStyle */
int     fill_rule;
/* Режим заполнения многоугольников */
int     arc_mode;
/* Режимы заполнения дуг - XSetArcMode */
Pixmap      tile;
Pixmap      stipple;
/* Значения закрасок для fill_style */
int         ts_x_origin;
int         ts_y_origin;      /* Сдвиги */
Font         font;                /* Шрифт */
int          subwindow_mode;
/* Режим влияния подокон - XSetSubwindowMode */
Bool         graphics_exposures;
/* Запросы на перерисовку */
int          clip_x_origin;
int         clip_y_origin;    /* Отсечения */
Pixmap      clip_mask;        /* Пикселы для вывода */
int         dash_offset;      /* Сдвиг пунктира */
char        dashes;            /* Вид пунктира */
} XGCValues ;
```

Из всего этого нам надо уметь только выводить строки — процедурами XDrawString, XDrawText, XDrawString16, XDrawText16, XDrawImageString, XDrawImageText, XDrawImageString16, XDrawImageText16, причем процедуры, оканчивающиеся на «16», используются для длинных шрифтов, а остальные для коротких.

Для загрузки шрифта используется вызов XLoadQueryFont. Например, так:

```
XFontStruct *prFont;
....
/* Загрузка */
if (( prFont=XLoadQueryFont(prDisplay,
```

```
"*-charter-*" )) == NULL )
sys_err("Font not found !");
```

Имя шрифта состоит из двух частей, разделенных «—», причем первая часть шрифта представляет строку вида «-изготовитель-имя-жирность-наклон-ширина», а вторая — «размер в пикселах - размер в точках - горизонтальное разрешение вертикальное разрешение - тип - средняя ширина - множество символов-», причем части могут заменяться знаками «*» и «?».

Собственно говоря, на этом можно и закончить разбор программы. Я хочу извиниться, если кому-то объяснение показалось слишком подробным, очевидно, я просто писал не для вас. Все остальные могут набрать в системе приводимые далее команды, если, конечно, они не сделали предварительно makefile, и поиграть в «Тетрис».

```
g++ window.c client.c -o client -lX11 &
g++ window.c server.c -o server -lX11 &
```

По окончании компиляции, чтобы убедиться в работоспособности программы, вы можете запустить в XWindow эмулятор какого-нибудь терминала и набрать последовательно команды:

```
Server &
Client &
```

Потом, установив фокус на окно программы-клиента, наберите любой текст, и он появится в окне сервера. Я не пытался привести здесь законченную полезную программу, в сущности, это только идея, которую вы, возможно, захотите развить. Так что вперед, к новым вершинам, а о программировании с использованием Xt я постараюсь рассказать в другой раз. Я советую прочитать литературу, указанную далее, однако вы вполне обойдетесь и без этого, если будете пользоваться командой man.

Адрес автора:
karlson@sch57.mcn.msk.su (Никита Кожекин)

Литература

1. **Nye A. Xlib Programming Manual.— USA: O'Relly & Associates, 1988.**
2. **Nye A. Xlib Reference Manual.— USA: O'Relly & Associates, 1988.**

Листинг 1. MESSAGE.H

```
/* Включаемый модуль: message.h  */
#include <sys/types.h>
#include <sys/ipc.h>
#include <sys/sem.h>  /* Работа с семафорами */
#include <sys/shm.h>   /* Работа с разделяемой памятью */

/* Работа в XWindow */
#include <X11/Xlib.h>
#include <X11/Xutil.h>
#include <X11/Xos.h>
#include <X11/keysym.h>

#include <stdlib.h>
#include <stdio.h>
#include <string.h>

#define SEM_ID    2001   /* Ключ массива семафоров */
#define SHM_ID     2002  /* Ключ разделяемой памяти */
#define PERMIS    0666    /* Права доступа: все могут
                             читать и писать*/

#define MSG_TYPE_EMPTY   0 /* Типы сообщений:   */
#define MSG_TYPE_STRING  1 /* пустое, строка,   */
#define MSG_TYPE_FINISH  2 /* конец диалога     */

#define MAX_STRING                 20
 /* Максимальная длина сообщения */

#define WND_WDT          200 /* Ширина окна */
#define WND_HGH          100 /* Высота окна */
#define WND_MIN_WDT     50 /* Минимальная ширина окна */
#define WND_MIN_HGH     50 /* Минимальная высота окна */
#define WND_BORDER_WDT 5    /* Ширина бордюра */

struct msg_t {
    int type;
    char string[MAX_STRING];
}; /* Структура сообщения */

inline void sys_err( char * msg )
{
  puts( msg );
  exit( 1 );
} /* Системная ошибка */
/*  Работа со структурой semun */
inline union semun semun (int i)  {
  union semun s;
  s.val = i;
  return s;
}

inline union semun
         semun (struct semid_ds * b)
```

Листинг 1 (продолжение)

```
{
  union semun s;
  s.buf = b;
  return s;
}

/* Инициализация окна написана в window.c */
void SetWindowManagerHints (
        Display     * prDisplay,
        char        * psPrgClass,
        char        * argv[],
        int         argc,
        Window      nWnd,
        int         x ,
        int         y,
        int         nWidth,
        int          nHeight,
        int         nMinWidth,
        int          nMinHeight,
        char        * psTitle,
        char         * psIconTitle,
        Pixmap      nIconPixmap);
```

Листинг 2. WINDOW.C

```
/* Инициализация окна */

#include "message.h"

void SetWindowManagerHints (
    Display *  prDisplay,
    char    *  psPrgClass,
    char    *  argv[],
    int      argc,
    Window    nWnd,
    int      x,
    int      y,
    int      nWidth,
    int      nHeight,
    int      nMinWidth,
    int      nMinHeight,
    char    *  psTitle,
    char    *  psIconTitle,
    Pixmap    nIconPixmap)
{
  XSizeHints rSizeHints;
#ifdef X11R3  /* X11R3 и ниже */
  rSizeHints.flags        = PPosition | PSize | PMinSize;
  rSizeHints.x             = x;
  rSizeHints.y             = y;
  rSizeHints.width         = nWidth;
  rSizeHints.height        = nHeight;
  rSizeHints.min_width     = nMinWidth;
  rSizeHints.min_height    = nMinHeight;

  XSetStandardProperties ( prDisplay, nWnd, psTitle,
                    psIconTitle, nIconPixmap,
                    argv, argc, &rSizeHints );
#else /* X11R4 и выше */
  XWMHints      rWMHints;
  XClassHint    rClassHint;
  XTextProperty  prWindowName, prIconName;
  if ( !XStringListToTextProperty(&psTitle, 1,
       &prWindowName) || !XStringListToTextProperty(
       &psIconTitle, 1, &prIconName))  {
        sys_err("No memory!");
  }

  rSizeHints.flags        = PPosition | PSize | PMinSize;
  rSizeHints.min_width     = nMinWidth;
  rSizeHints.min_height    = nMinHeight;

  rWMHints.flags = StateHint | IconPixmapHint | InputHint;

  rWMHints.initial_state   = NormalState;
  rWMHints.input           = True;
  rWMHints.icon_pixmap     = nIconPixmap;

  rClassHint.res_name      = argv[0];
  rClassHint.res_class     = psPrgClass;

XSetWMProperties ( prDisplay, nWnd, &prWindowName, &prIconName,
                   argv, argc, &rSizeHints,&rWMHints, &rClassHint);
#endif
}
```

Листинг 3. CLIENT.C

```
/* Листинг программы клиента: client.c
Компиляция : g++ window.c client.c -o client -lX11 & */

#include "message.h"

#define WND_X          0 /* Координаты окна */
#define WND_Y        100

#define WND_TITLE      "Client Window"
/* Заголовок окна */
#define WND_ICON_TITLE  "Client Icon"
/* Заголовок пиктограммы */
#define PRG_CLASS      "Client"

int main(int argc, char * argv[])
/* Основная процедура */
{
 Display     *prDisplay;
 Window       nWnd;
 int          semid,shmid,x,y,nScreenNum;
 GC           prGC;
 XEvent       rEvent;
 msg_t        *msg_p;
 char         s[MAX_STRING];
 KeySym       nKeySym,naModList[2];

 if (( semid = semget(SEM_ID,1, 0)) < 0 )
  sys_err("CLIENT: can't get semaphore!");
  /* Чтение массива семафоров */
 if (( shmid = shmget(SHM_ID,sizeof(msg_t),0))<0)
  sys_err("CLIENT: can't get shared memory segment!");
 /* Доступ к сегменту разделяемой памяти */
 if ((msg_p = (msg_t * ) shmat(shmid, 0, 0)) == NULL)
  sys_err("CLIENT: shared memory attach error!");
  /* Получение адреса */

 if ((prDisplay = XOpenDisplay(NULL))==NULL)
   sys_err("CLIENT: can't connect to the X server!");
 /* Подключение к X-серверу */

/* Регистрация окна */
 nScreenNum = DefaultScreen(prDisplay);
 nWnd = XCreateSimpleWindow(prDisplay,
       RootWindow(prDisplay,nScreenNum), WND_X,
       WND_Y, WND_WDT, WND_HGH, WND_BORDER_WDT,
       BlackPixel(prDisplay, nScreenNum),
       WhitePixel(prDisplay,nScreenNum));
 SetWindowManagerHints(prDisplay, PRG_CLASS, argv,
                  argc, nWnd, WND_X, WND_Y,
                  WND_WDT, WND_HGH, WND_MIN_WDT,
                  WND_MIN_HGH, WND_TITLE,
                  WND_ICON_TITLE, 0);

/* Выбор чтения событий */
 XSelectInput(prDisplay, nWnd, ExposureMask | KeyPressMask);

 /* Показ окна */
 XMapWindow(prDisplay, nWnd);

 /* Установка новой кнопки клавиатуры */
 naModList[0] = XK_Control_L;
 naModList[1] = XK_Shift_L;
 XRebindKeysym( prDisplay, XK_A, naModList, 2, "EXIT"
              strlen("EXIT"));
 x = 0;
 y = 0;
 /* Создание графического контекста */
 prGC = XCreateGC(prDisplay, nWnd, 0, NULL);

 /* Главный цикл: чтение событий */
 for (;;) {
```

```
        /* Выбор события из очереди */
        XNextEvent(prDisplay, &rEvent);

   switch (rEvent.type) {
     case Expose : /* Запрос на перерисовку */
       if (rEvent.xexpose.count!=0) break;
       XSetForeground(prDisplay, prGC,
                      BlackPixel(prDisplay, 0));
       /* Вывод строки */
       XDrawString(prDisplay, nWnd, prGC, 10, 50,
                "Client,", strlen("Client,"));
       break;

     case KeyPress :
       /* Нажатие клавиши */
       memset(s, 0, sizeof(s));
       XSetForeground(prDisplay, prGC, BlackPixel(prDisplay, 0));
       XDrawString(prDisplay, nWnd, prGC, 10, 66,
                "TALK:", strlen("TALK:"));
      /* Чтение строки-соответствия нажатой клавиши */
       XLookupString(&rEvent.xkey, s, sizeof(s), &nKeySym, NULL);
       /* Ждем-с ... */
       while ( semctl (semid, 0, GETVAL, semun(0) ) !:
                    msg_p->type != MSG_TYPE_EMPTY);
       /* Блокировка */
       semctl(semid, 0, GETVAL, semun(1));
       strncpy(msg_p->string,s,MAX_STRING);
       /* Эхо-отображение */
       XDrawString(prDisplay, nWnd, prGC, 10+x, 80 + y, s, strlen(s));
       if ((10+x) > WND_WDT) {x =0; y=y+10;}
                     else x = x + 10;
       if (strcmp(s,"EXIT"))
            msg_p->type= MSG_TYPE_STRING;
            else msg_p->type = MSG_TYPE_FINISH;
      /* Снятие блокировки */
       semctl(semid, 0, SETVAL, semun(0));
       if (!strcmp(s,"EXIT"))  {
       /* Конец работы */
                     shmdt((char*)msg_p);
                     XFreeGC(prDisplay,prGC);
                     XCloseDisplay(prDisplay);
                     exit(0);
       }
       break;
     }
   }
   /* Освобождение графического контекста */
   XFreeGC(prDisplay,prGC);
   XCloseDisplay(prDisplay);
   exit(0);
}
```

Листинг 4. SERVER.C

```
/* Листинг программы сервера: server.c
Компиляция : g++ window.c server.c -o server -lX11 & */

#include "message.h"

#define WND_X 300 /* Координаты */
#define WND_Y 100

#define WND_TITLE          "Server Window" /* Заголовки */
#define WND_ICON_TITLE     "Server Icon"
#define PRG_CLASS          "Server"

int main(int argc, char * argv[])
{
  Display     *prDisplay;
  Window      nWnd;
  int          semid,shmid,x,y,nScreenNum;
  GC          prGC;
  XEvent      rEvent;
  msg_t       *msg_p;
  char        s[MAX_STRING];

  /* Создание семафоров */
  if ((semid = semget(SEM_ID,1, PERMIS : IPC_CREAT)) < 0 )

  sys_err("SERVER: can't create semaphore!");

  /* Создание разделяемого сегмента */
  if ((shmid = shmget(SHM_ID,sizeof(msg_t),
                      PERMIS : IPC_CREAT))<0)
  sys_err("SERVER: can't create shared memory segment!");
  if ((msg_p = (msg_t * ) shmat(shmid, 0, 0)) == NULL)
  sys_err("SERVER: shared memory attach error!");
  semctl( semid, 0 , SETVAL, semun(0));
  msg_p->type = MSG_TYPE_EMPTY;

  /* Подключение к X-серверу */
  if ((prDisplay = XOpenDisplay(NULL))==NULL)
  sys_err("SERVER: can't connect to the X server!");

  /* Инициализация окна и экрана */
  nScreenNum = DefaultScreen(prDisplay);
  nWnd = XCreateSimpleWindow(prDisplay,
        RootWindow(prDisplay, nScreenNum), WND_X,
        WND_Y, WND_WDT, WND_HGH, WND_BORDER_WDT,
        BlackPixel(prDisplay, nScreenNum),
        WhitePixel(prDisplay, nScreenNum));
  SetWindowManagerHints(prDisplay, PRG_CLASS, argv,
                  argc, nWnd, WND_X, WND_Y,
                  WND_WDT, WND_HGH, WND_MIN_WDT,
                  WND_MIN_HGH, WND_TITLE,
                  WND_ICON_TITLE, 0);

  /* Выбор событий: перерисовка, клавиатура */
  XSelectInput(prDisplay, nWnd,
                 ExposureMask : KeyPressMask);

  /* Вывод окна */
  XMapWindow(prDisplay, nWnd);
  x = 0;
  y = 0;
  /* Создание графического контекста */
  prGC = XCreateGC(prDisplay, nWnd, 0 ,NULL);
  for (;;) {
    if (msg_p->type != MSG_TYPE_EMPTY) {
       /* Блокировано - ждем-с ... */
       if (semctl(semid, 0, GETVAL, semun(0)))  continue;
       /* Блокировка */
       semctl(semid, 0, SETVAL, semun(1));
       if (msg_p->type == MSG_TYPE_STRING) {
         XSetForeground(prDisplay, prGC, BlackPixel(prDisplay, 0));
         /* Вывод прочитанной строки */
         XDrawString(prDisplay, nWnd, prGC, 10 + x,
                     78 + y, msg_p->string, strlen(msg_p->string));
         if ((10+x) > WND_WDT) {x =0; y=y+10;}
                        else x = x + 10;
       }
       if (msg_p->type == MSG_TYPE_FINISH) break;
       msg_p->type = MSG_TYPE_EMPTY;
       /* Отмена блокировки */
       semctl(semid, 0, SETVAL, semun(0));
    }
     /* Вывод приглашения для тех, кто понял */
     XSetForeground(prDisplay, prGC, BlackPixel(prDisplay, 0));
     XDrawString(prDisplay, nWnd, prGC, 10, 50,
                 "Server,",strlen("Server,"));
     XDrawString(prDisplay, nWnd, prGC, 20, 66,
                 "LISTEN:",strlen("LISTEN:"));
  }

/* Освобождение графического контекста */
  XFreeGC(prDisplay, prGC);
  /* Закрытие дисплея */
  XCloseDisplay(prDisplay);
  if (semctl(semid, 0, IPC_RMID, semun(((struct semid_ds *) 0)) )<0)
  sys_err("SERVER: semaphore remove error!");
  /* Удаление массива семафоров */
  shmdt((char *)msg_p);
  /* Удаление сегмента разделяемой памяти */
  if ( shmctl( shmid, IPC_RMID, (struct shmid_ds *) 0)<0)
  sys_err("SERVER: shared memory remove error!");
  /* Выход */
  exit(0);
}
```

Некоторые из Ваших серьезных конкурентов уже используют Power Macintosh.

•

Приятных сновидений.

Они используют мощный RISC-процессор. Они работают с программами Макинтоша, DOS и Windows.*

Они больше успевают. Возможно, они больше зарабатывают. Подумайте об этом. Позвоните 978-4262 или 978-1257. Спросите телефон Вашего дилера Apple. Не теряйте время.

Apple

Power Macintosh уже в Москве. Чего же Вы ждете?

**Используя программное обеспечение SoftWindows™ фирмы Insignia Solutions Inc*

® RUI IMC Apple Computer Москва 103055, 2-й Вышеславцев Пер., д. 17 Тел. 978-1257/4262/2821/3562/2100, факс 978-1391. Сделано на компьютере Apple Power Macintosh.

Система программирования Stony Brook Pascal+

А. Тихонов

В статье рассмотрена система программирования Stony Brook Pascal+ (SBP), включающая в себя интегрированную среду и оптимизирующий компилятор, входной язык которого полностью соответствует Turbo Pascal версии 6.0. SBP создает стандартные объектные модули, имеются интерфейсы для сопряжения с Borland и Microsoft C.

ПРОГРАММЫ

Рассматривая любую систему программирования на Pascal для IBM PC, в качестве точки отсчета приходится выбирать Turbo (TP), а с 1992 года Borland Pascal (BP) фирмы «Borland International, Inc.», ставший стандартом де-факто, обладающий сверхвысокой скоростью компиляции и удобной интегрированной средой разработки программ.

Платой за высокую скорость компиляции является не слишком высокое качество генерируемого кода, а за интеллектуальные и безопасные средства организации межмодульных интерфейсов — нестандартные форматы объектных модулей .TPU и библиотек .TPL, меняющиеся от версии к версии и несовместимые между собой.

Последняя особенность делает невозможным использование библиотек, наработанных на Turbo и Borland Pascal, с другими системами программирования в среде MS-DOS. Обратное — использование стандартных объектных модулей при программировании на TP возможно, но требует хорошего знания внутренней «кухни» системы программирования, описанной в недавно появившемся на российском рынке продукте Borland Open Architecture for Pascal [1]. Надо сказать, что под Windows и в защищенном режиме (при использовании DLL) проблема межъязыковых интерфейсов снимается.

Появление системы программирования Pascal+ американской фирмы «Stony Brook Software» во многом напоминает ситуацию, сложившуюся с системой программирования PL/1, когда фирма IBM была вынуждена разработать парные компиляторы PL/1 CHECKOUT и PL/1 OPTIMIZER. Первый из них обладал высокой скоростью компиляции, мощными средствами отладки, но ограниченной совместимостью с другими системами программирования. PL/1 OPTIMIZER предназначался для создания «боевых» программ.

Stony Brook Pascal+ (SBP) является «ортогональным дополнением» к Borland Pascal и существенно расширяет возможности использования Pascal при создании профессионального программного обеспечения в среде MS-DOS. Основные свойства системы программирования SBP рассмотрены в [2], поэтому в данной статье сделан упор на ее тестирование.

Общая характеристика SBP

SBP поставляется на трех дискетах высокой плотности и включает оптимизирующий компилятор, редактор связей, библиотекарь, отладчик, профайлер, интегрированную систему разработки, два программных пакета для создания текстовых и графических интерфейсов. Документация состоит из двух книг, первая их которых посвящена входному языку и системе программирования, а вторая описанию библиотеки, практически полностью совпадающей с TP.

Анализируемая версия продукта 6.1 выпущена в марте 1992 года и по входному языку, как заявлено в документации и рекламных материалах фирмы, полностью соответствует версии 6.0 Turbo Pascal. Печатная документация соответствует предыдущей версии 6.0 SBP, все отличия и дополнения, появившиеся в версии 6.1, представлены в документах, поставляемых на магнитных носителях.

SBP почти полностью написан на языке программирования Modula-2 разработки «Stony Brook

Software», имеет общий с нею генератор кода и в этом отношении похож на TopSpeed Pascal [3].

Совместимость с BP

Тестированию подвергалась программа для просмотра графических файлов в формате GIF (обозначим ее «программа 1»). Для анализа работы SBP с вычислительными задачами была выбрана программа расчета электронных спектров аморфных материалов, любезно предоставленная В.А. Лигачевым (обозначим ее «программа 2»), которая осуществляет решение проблемы собственных значений для больших матриц и использует Turbo Vision. Третьей программой, использованной для тестирования больших проектов, являлось САПР приложение объемом около 30 тысяч строк исходного текста, написанное при участии автора в 1991 году (обозначим «программа 3»).

Программа 1 была оттранслирована с первого захода. Трудности с перетрансляцией второй программы, связанные с ассемблерными модулями Turbo Vision, были преодолены после прочтения документа, поставляемого с SBP на магнитных носителях и специально посвященного данному вопросу.

В третьей программе использовалась первая версия пакета TEGL Windows Toolkit фирмы «TEGL Systems Corp.», предназначенного для построения графических интерфейсов. Все изменения, внесенные в исходный текст программы, определялись отличиями второй версии TEGL, поставляемой в составе SBP, от первой.

В дополнение к перечисленным тестовым задачам с помощью SBP были перекомпилированы еще 18 разнородных программ общим объемом 14 тысяч строк исходного текста. Отличий по входному языку между Turbo Pascal и SBP обнаружено не было.

Отметим, что входной язык SBP является расширением входного языка Turbo Pascal, связанного с дополнительными опциями и описателями, используемыми при оптимизации программ и для межъязыковых интерфейсов. Скорость компиляции SBP много ниже, чем у BP. Средний коэффициент замедления составлял от 16 до 20 при работе с SBP в защищенном режиме. Это привело к тому, что программа 3 компилировалась и редактировалась на ПЭВМ 486DX 33 МГц более десяти минут. В связи с этим у SBP отличная от BP стратегия обработки ошибок. BP останавливает компиляцию при обнаружении первой ошибки, а SBP при обнаружении ошибки продолжает компиляцию, позволяя после одного запуска компилятора выявить и исправить до 25 ошибок. Информация об ошибках у SBP более содержательная, чем у TP.

Как упоминалось ранее, в комплект поставки SBP включены отладчик и профайлер. Функционально они эквивалентны соответствующим компонентам TP. Отладочная информация генерируется как для работы с собственным отладчиком SBP, так и в стандартном формате, совместимом с MS CodeView. Назначение функциональных клавиш отличается от встроенного отладчика BP, так что отлаживаться в BP привычнее.

ПРОГРАММЫ

Оптимизирующий компилятор Stony Brook Pascal+ полностью совместим с Turbo Pascal 6.0 по входному языку и создает объектные модули в стандартном формате OBJ.

Редактор связей

Уникальным свойством SBP является компиляция исходного текста не только в объектные модули, но и непосредственно в библиотеку объектных модулей, соответствующих проекту. Запуск библиотекаря осуществляется непосредственно из интегрированной системы.

Каждому модулю исходного текста (program, unit) соответствует объектный модуль. Объектный модуль включается редактором связей в исполняемый модуль целиком. При работе с библиотеками редактор связей включает только те процедуры и функции, которые используются в программе. То же касается статических и частично виртуальных методов. В документации приводятся условия, когда выборочное (smart) редактирование связей для виртуальных методов необходимо отключить.

Создание библиотек объектных модулей значительно ускоряет процесс редактирования связей, но их не рекомендуется использовать в процессе отладки, так как размер библиотеки может превышать суммарный объем объектных файлов в пять и более раз из-за необходимости записывать отладочную информацию в объектный код каждой процедуры и функции, а не модуля.

Оптимизация

Pascal был разработан Н. Виртом прежде всего для целей обучения, и исторически сложилось так, что компиляторы для него, отличаясь высокой скоростью, ничего не делают для оптимизации генерируемого кода. Основной заслугой разработчиков SBP является возможность глубокой оптимизации программ при сохранении совместимости с TP. В SBP включено более 20 видов оптимизации как на высоком, так и на низком уровне, например, использование регистров процессора. Возможно разворачивание процедур и функций вместо их вызовов с помощью указания опции inline. Уникальным является разворачивание только в случае единственного вызова процедуры.

Компилятор позволяет раздельно включать и выключать каждый из видов оптимизации как из интегрированной среды разработчика, так и с помощью опций в исходном тексте программы. Указывать опции можно как глобально (для всего проекта), так и для отдельного модуля. В большинстве случаев вполне достаточно включения оптимизации по размеру исполняемого модуля или скорости выполнения. Различия при выборе типа оптимизации весьма существенны и иллюстрируются таблицей.

Для тестируемых программ оптимизация позволяет ускорить выполнение в 1,2...2,1 раза по сравнению с TP. Особенно большой эффект достигался при эмуляции вычислений с плавающей точкой на ПЭВМ без математического сопроцессора (программа 2). При оптимизации по размеру объем исполняемого модуля уменьшался в 1,2...1,4 раза.

Наличие оптимизирующего компилятора может во многом изменить стратегию создания программ. Стан-

дартным подходом является разработка приложения на языке программирования с последующим выявлении «узких» мест с помощью профайлера и переписывания их на Assembler. Оптимизация позволяет сократить эту часть работы. Одной из тестируемых программ являлся пакет компрессии/декомпрессии текстов «на лету», который был разработан на Turbo C для включения в программы на PDC Prolog, а затем почти полностью переписанный на Assembler. Переработка пакета на SBP дал всего на 10% худшие по скорости результаты по сравнению с вариантом на Assembler. Оптимизация по скорости дала объектный модуль, превышающий объем варианта на Assembler в 1,4 раза.

Перефразируя высказывание американских защитников свободной продажи стрелкового оружия, можно сказать, что оптимизирует не компилятор, а программист. Действительно, успех зависит от удачного выбора алгоритма, аккуратного кодирования, а также использования экономных языковых конструкций, большинство из которых реализовано в SBP более эффективно, чем в TP, например работа с множествами. Еще больший выигрыш можно получить при работе с множествами, представляемыми одно-, двух- и четырехбайтовыми словами. Поэтому оптимальный результат будет получен с типом

```
TYPE set15 = set of [0..31];
```

а не с

```
TYPE set15 = set of [1..32];
```

Дополнительные возможности оптимизации вручную дает опция генерации кода на Assembler, сопровождаемого исходным текстом программы. Этот код может быть переписан, ассемблирован и использован в проекте в виде объектного модуля.

Перед разработчиками оптимизирующих компиляторов всегда стоит дилемма: либо обеспечить почти полную безопасность генерируемого кода за счет переусложнения алгоритмов компиляции, либо переложить все заботы на программиста. При этом последнему рекомендуется в том случае, если некий модуль программы функционирует ненадлежащим образом, отключить его оптимизацию, что приводит к необходимости повторного тестирования программы при оптимизации.

Разработчики SBP пошли по разумному пути, введя дополнительные описатели, обеспечивающие компилятор дополнительной информацией и позволяющие последнему сгенерировать корректный код, не отключая оптимизации. Так, необходимо явно указывать с помощью описателя volatile на асинхронно (например с помощью прерываний) изменяемые переменные или переменные, имеющие псевдонимы (описатель alias). Таким образом с помощью разумного компромисса понижается сложность оптимизации при обеспечении высокой степени безопасности генерируемого кода.

Межъязыковые связи

В SBP хорошо продуманы вопросы поддержки разработки сложных программных систем, содержащих разноязычные модули. Делается это на нескольких уровнях: во-первых, с помощью гибкой системы адресации, позволяющей указывать модели памяти раздельно для кода, данных и констант, во-вторых, за счет большого числа опций, управляющих межмодульными интерфейсами. И, наконец, интегрированная среда позволяет управлять проектами, содержащими модули на SBP, C и Assembler.

SBP позволяет легко интегрировать свои объектные модули в программы, созданные на других языках программирования, и использовать библиотеки объектных модулей третьих фирм, разработанные на C и Assembler, в программах на Pascal.

В качестве примера приведем коротенькую программу перекодировки строк, C-аналог которой используется в программах на PDC Prolog для перевода строк в верхний регистр, шифровки и так далее. Интерфейс для нее на Prolog имеет вид (пример 1).

Известно, что в программе может быть только один эмулятор операций с плавающей точкой. SBP позволяет либо использовать собственный эмулятор, либо отключить его, используя эмулятор, например C.

Интегрированная среда SBP

Интегрированная среда (ИС) SBP функционирует как в реальном, так и защищенном режиме и построена вокруг так называемой библиотеки проекта. В ней

Пример 1

```
GLOBAL PREDICATES
  RECODE(STRING,STRING) - (i,i) LANGUAGE C AS "_RECODE"
% RECODE(СТРОКА_ДЛЯ_ПЕРЕКОДИРОВКИ, ТАБЛИЦА_ПЕРЕКОДИРОВКИ)
Модуль на SBP представляет собой следующее:
UNIT CODE;
     {$//ALIGN:1} {Выравнивание данных PDC PROLOG осуществляется
                   на границу байта. По умолчанию в SBP
                   выравнивание на границу слова.
                   Синтаксис задания расширенных опций
                   отличается от BP}
{$//NOMODINIT}{Отключение генерации кода инициализации модуля
{$//CODE:L//DATA:L//CONST:L//NOOPT:X}{Указание модели памяти
                   large для кода, данных и констант}
{$M 2048,0,0}  {Отключение системы управления памятью SBP}
{$//STACK:DS}  {Стек адресуется через регистр DS}
{$A-}
INTERFACE
   TYPE
        BYTEARRAY = ARRAY[0..65535] OF BYTE; {Представление
                                    перекодируемой строки}
        TABLE     = ARRAY[1..255] OF BYTE;    {Представление
                                    таблицы перекодировки }
   PROCEDURE RECODE(VAR S:BYTEARRAY; VAR T:TABLE) [C]
                          PUBLIC '_RECODE'; {Таким образом
                          задается соглашение о передаче
                          параметров для C и внешнее имя
                          процедуры}
IMPLEMENTATION
   PROCEDURE RECODE(VAR S:BYTEARRAY; VAR T:TABLE) [C]
                                    PUBLIC '_RECODE';
      VAR
         I: WORD;
         B: BYTE;
       BEGIN
         I:=0;
         WHILE S[I] < >0 DO BEGIN
             S[I] := T[S[I]]; INC[I];
         END;
       END;{RECODE}
END. {CODE}
```

■

хранится информация об исходных, объектных, исполняемых модулях проекта, файлах описания редактора связей, а также их взаимных зависимостях. Допускается включение в проект модулей на других языках программирования. При необходимости ИС запускает для них компиляторы, для чего нужно задать соответствующие командные строки. Возможна иерархическая организация проекта, то есть включение в него других библиотек проектов. Например, при использовании библиотеки объектных модулей необходимо для нее создать интерфейсный модуль и библиотеку проекта, которую можно далее включать в другие проекты.

Как ни странно, в ИС SBP вести разработку больших проектов гораздо удобнее, чем небольших программ, состоящих из одного-двух модулей. Предполагается, что каждому проекту соответствует отдельная директория. При первом запуске SBP в ней создается библиотека проекта и две поддиректории — SOURCE и OBJ — для исходных и объектных модулей соответственно. Модули проекта должны быть вручную включены в библиотеку проекта. В ИС отсутствует возможность просмотреть оглавление и задать в нем имя файла, как это сделано в TP. Поэтому для упрощения тестирования я написал утилиту, которая создавала директории и копировала в них файлы по списку. Далее, запустив SBP, я в один прием включал в библиотеку проекта все файлы из поддиректории SOURCE.

Как уже говорилось, в ИС SBP легко интегрируются внешние программы. Так, допускается использование различных текстовых редакторов для работы с текстом программы и ошибками. Не вызывает трудностей вызов из ИС отладчика и профайлера. Существенным недостатком ИС является отсутствие интерактивной помощи по языковым конструкциям, стандартным процедурам и функциям, которая уже давно стала привычной для TP.

Тестируемая версия SBP не позволяет создавать программы, функционирующие в защищенном режиме, зато для создания больших программных систем поддерживаются две системы оверлеев. Первая из них позволяет работать со статическими оверлеями в стиле PLINK86 и обеспечивается собственным редактором связей SBP, а вторая — динамическая [5]. Уникальным свойством оверлеев SBP является наличие описаний используемых структур данных, что позволяет, например, написать средства для сжатия исполняемых модулей с оверлейной структурой.

Windows и разработка графических приложений

SBP позволяет вести разработку Windows программ двумя способами. Для редактирования связей необходимо наличие Microsoft LINK.EXE, который автоматически вызывается интегрированной средой, но не входит в комплект поставки.

Для создания Windows приложений, использующих Windows API, необходимо наличие MS Windows SDK, однако большинство программ можно перетранслировать и выполнить в среде Windows без каких-либо изменений: достаточно запустить интегрированную среду с опцией /W, поскольку «Stony Brook Software» разработала специальную библиотеку, реализующую в среде Windows практически всю run-time библиотеку BP, включая BGI-графику. Естественными ограничениями являются невозможность работы с прерываниями, абсолютными адресами памяти, страницами видеопамяти, палитрой EGA/VGA. Такой перенос DOS-программ в среду Windows, естественно, не решает всех вопросов, но в ряде случаев может помочь.

В качестве тестового примера мы использовали программу, рассмотренную в [4] и представляющую средство просмотра картинок в формате GIF. Было решено сконструировать исполняемый модуль, автоматически распознающий среду и выполняющийся как под MS-DOS, так и Windows.

В отличие от TopSpeed Pascal [3] программа откомпилировалась с первой попытки, но запустить ее я не смог. При просмотре документации, поставляемой на магнитных носителях, было обнаружено, что BGI-графика поддерживается аж двумя способами. В первом случае с помощью TP необходимо скомпилировать и отредактировать небольшой загрузочный модуль, который подгружается каждой программой, использующей BGI, и обрабатывает все вызовы BGI. Этот не слишком изящный подход использован, видимо, для того чтобы не покупать лицензии на BGI у «Borland».

Гораздо проще оказался второй подход, полностью имитирующий BGI с помощью технологии, приобретенной у «TEGL Systems Corp.». Заключается он в том, что вместо драйверов .BGI используются драйверы с расширением .TGI. Скопировав драйвер VGA в директорию с программой, я получил результаты, приведенные в таблице. Последующее тестирование других имеющихся у меня графических программ не выявило отклонений в функционировании.

Для получения исполняемого модуля, работающего как в MS-DOS, так и в Windows, необходимо сначала скомпилировать модуль для DOS, получивший название DOSGIF, а после перетрансляции его для среды Windows немного модифицировать файл определений для редактора связей:

```
NAME DOSWIN
EXETYPE WINDOWS
STUB 'DOSGIF.EXE'
DATA MOVEABLE
HEAPSIZE 1024
STACKSIZE 16384
EXPORTS CrtWindowProc
EXPORTS BgiWindowProc
```

Добавление строчки STUB 'DOSGIF.EXE' приводит к тому, что при редактировании связей исполняемые модули для MS-DOS и Windows объединяются и при запуске программы в среде MS-DOS управление автоматически передается модулю DOSGIF. Перед запуском такую объединенную программу необходимо пометить как Windows-программу с помощью входящей в комплект поставки утилиты MARKWIN. Кроме этого, отметим, что программы, перенесенные под Windows, работают медленнее, чем в среде DOS (фактор замедления для текстовых приложений менялся от 1,2 до 1,8), а скорость работы BGI-графики вообще удручающая (см. таблицу).

Средства поддержки программирования BSP в среде Windows являются стандартными для систем программирования первого поколения: работа компилятора в среде DOS, наличие файла описания процедур Windows API и MS Windows SDK, отсутствие визуальных средств программирования. Особенностью же является возможность переноса уже имеющихся программ в среду Windows.

Таблица

	Размер загрузочного модуля в байтах	Время выполнения, с
Программа, оттранслированная BP	28488	12,4
Программа, оттранслированная SBP:		
оптимизацияразмера программы	21974	7,2
оптимизациявремени выполнения	30780	7,0
Программа, оттранслированнаяSBP для MS-DOS/Windows	56319	7,2 (DOS)/55 (Windows)

Дополнительные средства

В комплект поставки SBP входят два продукта для построения пользовательских интерфейсов, первый из них — Object Toolkit фирмы «TechnoJock Software» для работы в текстовом режиме, второй — TEGL Windows Toolkit фирмы «TEGL Systems Corp.» предназначен для построения графических интерфейсов.

Object Toolkit поставляется в исходных текстах с полной документацией и по идеологии ближе к аналогичным средствам Object Professional, а не Turbo Vision (TV) фирмы «Borland». Если вы еще не привыкли к раскладкам экранов и внешнему облику Turbo Vision, то имеет смысл поэкспериментировать с Object Toolkit, тем более, что его архитектура много проще и прозрачнее, чем у TV, и гораздо легче поддается модификации.

TEGL Windows Toolkit (TWT) поставляется в ограниченном виде, содержит полный набор объектных модулей и исходные тексты примеров использования пакета. Хочется сказать несколько теплых слов о TWT. Разработан он семейной фирмой, в которой муж программировал и заодно присматривал за детьми, а жена занималась маркетингом и обработкой почтовых заказов. TWT поддерживается для наиболее распространенных компиляторов C и Pascal. Вместе с TWT поставляются средства для разработки компьютерных игр. Последнее определяет основное свойство TWT: работает он очень быстро. TWT поддерживает собственные графические примитивы, совместимые, в частности, с BGI, большое число графических адаптеров, очень быстрые средства для работы с растровыми шрифтами. Вместе TWT поставляется большое число шрифтов, в полной версии имеется дизайнер шрифтов, позволяющий конструировать, например, шрифты кириллицы.

В отличие от Windows, где осуществляется перерисовка объектов окон, TWT сохраняет образы окон в памяти, для чего используется собственная подсистема виртуальной памяти, которую можно приспособить для хранения и других данных.

TWT достаточно прост в освоении, программисту достаточно инициализировать окна, элементы управления и написать обработчики прерываний клавиатуры и «мыши». К сожалению, в TWT отсутствуют средства визуального программирования.

Предыдущая версия TWT очень сильно выручила, когда в 1991 году мне на пару со студентом в течение трех месяцев пришлось разработать достаточно объемное графическое приложение (программа 3). Следует отметить, что при этом нам пришлось исправить около десятка серьезных, хотя и достаточно очевидных ошибок в исходных текстах программ, реализующих элементы управления.

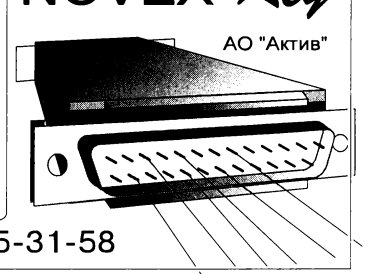
Заключение

Резюмируя сказанное, можно сказать, что SBP является инструментальной системой для создания профессиональных приложений на Pascal, полностью совместимой с TP по входному языку, с мощными средствами оптимизации, продуманными возможностями взаимодействия с другими системами программирования на основе стандартных библиотек объектных модулей. Таким образом «Stony Brook Software» сделала то, что уже давно следовало бы сделать «Borland» для обеспечения разработки профессионального программного обеспечения на Pascal. Остается надеяться, что несмотря на повышение стоимости разработки систем программирования, вытесняющего небольшие фирмы с рынка, следующие версии SBP будут соответствовать новым опциям и изменениям входного языка BP и позволят создавать оптимизированные программы для MS-DOS, защищенного режима и Windows. ■

Редакция благодарит фирму «Родник-СОФТ» за предоставленный для тестирования компилятор Stony Brook Pascal+.

Литература

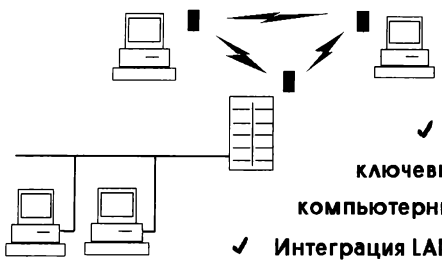
1. **А. Фищук. Borland Open Architecture for Pascal// SoftReview. — 1994. — № 3.**
2. **Ю. Кириллов. Паскаль-компилятор фирмы «Stony Brook Software»//SoftReview. — 1994. — № 1.**
3. **А. Тихонов. TopSpeed Pascal// Монитор. — 1993. — № 7-8.**
4. **А. Тихонов. Показать картинку. Часть 2// Монитор. — 1993. — № 5.**
5. **А. Тихонов. Вместить невместимое// Монитор. — 1994. — № 4.**

Электронные ключи
NOVEX Key
✔Защита программ и данных для DOS, Windows
✔Защита в Сетях
✔1000 бит энергонезависимой памяти
◆ Скидки от 5 шт.
◆ Цена при партии 500 шт. - 8.0 $
АО "Актив"

246-40-66 245-31-58

ELVIS+

Элвис +

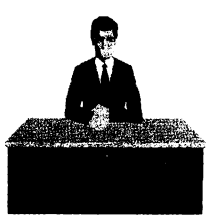
103460 Москва, Центральный проспект 11
Телефон : (095) 531-46 33, (095) 531-95-51
Факс : (095) 531-24-03
E-mail : sasha@elvis.ru, avs@elvis.ru



- ✓ Разработка ключевых элементов компьютерных радиосетей
- ✓ Интеграция LAN и WAN с использованием радиоканалов
- ✓ Радиодоступ к серверам Internet

- ✓ Интеграция и развитие сетевых технологий **Sun Microsystems**
- ✓ Обеспечение информационной безопасности распределенных корпоративных сетей

Sun



Консалтинг:

- ✓ по информационному сервису в Internet
- ✓ по созданию WWW-серверов
- ✓ по подключению к Internet
- ✓ по Sun/SPARC-технологиям
- ✓ по созданию и реконструкции корпоративных гетерогенных сетей: от IBM PC до отказо-устойчивых систем Tandem

WWW

- ✓ Сервис в сети Internet (WWW/Mosaic,E-mail, факс-сервис телеконференции) на WWW сервере Элвис+ < http://www.elvis.ru >

- ✓ Электронные издания компьютерных и коммерческих журналов
- ✓ Коммерческие базы данных

- ✓ Системы видео-конференций - "под ключ"
- ✓ Внедрение multi-media и hyper-media технологий

совместно с АО НТЦ Модуль и АООТ Вымпелком

- ✓ **"Мобильный офис"** дистанционное управление бизнесом
- ✓ Доступ к Internet-сервисам в среде сотовой системы BeeLine

ELVIS +

Centralny prospect 11, Moscow, Zelenograd
Russia 103460
Phone: (095) 531-46-33, (095) 536-95-51
Fax: (095) 531-24-03
E-mail: sasha@elvis.ru, avs@elvis.ru

Процедурные файлы — платформа для реализации полиморфизма в среде xBase

А. Дубина,
С. Сынгур

ТЕХНОЛОГИИ

В статье предлагается реализация полиморфного вызова методов объектов в среде языков xBase.

От редактора

Уважаемый читатель!

Ни для кого не секрет, что увеличение процессорной мощности персональных компьютеров привело к смещению интереса от систем управления базами данных, работающих в среде DOS, к их аналогам, функционирующим в среде Windows. Об этом свидетельствует и редакционная почта. Распространение на рынке таких инструментальных средств для Windows, как MS Access, MS Fox Pro, Paradox, CA-Clipper и версий СУБД, поддерживающих технологию «клиент-сервер», безусловно, окажет влияние на тематику нашего журнала. Это не означает, что DOS забыта, однако некоторые статьи про СУБД под DOS будут помещаться в «Монитор-Диск». Так, в «Монитор-Диск» № 1 за 1995 год помещены статьи А. Ермолова «Как организовать подсказку в приложении Paradox 3.5 и еще кое о чем» и «Paradox 4.0. Первое знакомство».

Невозможность применения принципов объектно-ориентированного программирования (далее ООП) при использовании языков 4GL не является столь критичным недостатком, как в случае языков 3GL. Однако использование ООП даже в усеченном виде (реализованного средствами самих этих языков, без дополнительных надстроек) весьма целесообразно, так как позволяет в некоторой степени повысить «прозрачность» ваших программ, уменьшить объем исходных текстов и облегчить модификацию.

В [1] была предложена одна из реализаций псевдополиморфизма на уровне возможности вызова методов классов (а не методов объектов классов!), основанная на формировании имен процедур путем сложения имен классов и имен методов (с последующим вызовом процедур по макроподстановке).

В [2] рассмотрена реализация псевдополиморфизма с использованием набора процедурных файлов, каждый из которых

соответствует некоторому классу и содержит в себе процедуры-методы этого класса. Однако сделано это было на довольно «топорном» уровне. В этой статье дается развитие идей, изложенных в [2], и предлагается более изящная реализация — оболочка, делающая более удобным использование полиморфизма уже на уровне вызовов методов объектов классов.

Исходные тексты даны на языке FoxPro v2.0, однако в целом методика может быть использована на любом языке xBase, начиная с dBaseIII, за исключением того, что вместо конструкции

```
do <имя процедуры> in <имя процедурного файла>
```

в языках dBaseIII, dBaseIII+, FoxBase (любой версии), FoxPro v1.xx вам придется использовать конструкцию:

```
set procedure to <имя процедурного файла>
do <имя процедуры>
```

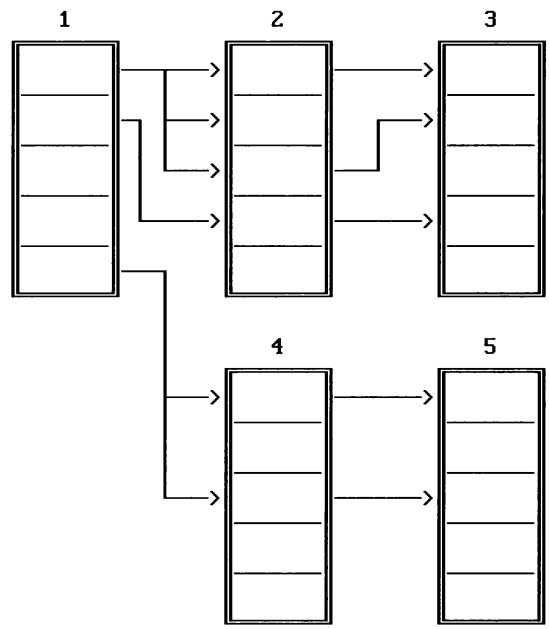


Рис. 1

До перехода к изучению листингов необходимо рассмотреть несколько принципиальных положений.

1. Никакие логические структуры, соответствующие классам или объектам классов, в памяти ПК или на диске не создаются.
2. Предполагается, что в системе определена возможность работы с различными объектами, причем они могут относиться к различным классам, которые объединяют объекты с одинаковыми наборами характеристик и действий (одинаковых действий!).
3. Под объектом некоторого класса понимается конкретная взаимосвязанная логическая структура, описанная просто на бумаге (например, объекту «бригада № 1», который относится к классу «бригады», соответствуют связанные записи в БД 1-2-3, объекту «цех № 1», который относится к классу «цехи», соответствуют связанные записи в БД 1-4-5, и так далее, рис. 1).
4. Каждый процедурный файл соответствует некоторому определенному вами классу, причем имена файла и класса полностью совпадают (отсюда следует первое ограничение по сравнению с другими ООП системами — длина имени класса не более 8 символов, допустимых в именах файлов).
5. Внутри процедурного файла находятся процедуры, имена которых являются и именами методов этих классов, и (очень важно!) в разных процедурных файлах имеются процедуры с одинаковыми именами, соответствующие общим логически однородным действиям (методам) объектов, например:

   calcwrk — расчет выработки (бригады, цеха, участка);

   timest — расчет суммарного времени простоя (бригады, цеха, участка).

   Очевидно, что эти процедуры должны быть одинаковыми (в простейшем случае) для всех объектов одного класса, но разными (в большинстве случаев) для объектов разных классов. Их единое логическое назначение делает крайне целесообразной возможность придания им одинаковых имен, что невозможно при процедурном программировании (традиционно пришлось бы иметь примерно такие имена процедур: calcwrk1, calcwrk2, calcwrk3, timest1, timest2, timest3).
6. Имена объектов, относящихся к разным классам, хранятся в простейшем случае в одной БД, в которой обязательно имеется символьное поле длиной 8 символов, предназначенное для хранения имени класса, к которому относится экземпляр, сведения о котором (в первую очередь имя экземпляра) находятся в соответствующей записи этой БД. Если же имена объектов содержатся в различных БД, то в каждой из них должно быть такое поле. В этой же базе (базах) могут быть поля, содержащие и другие сведения, структура которых одинакова для объектов различных классов, например дата регистрации системой или дата последнего обращения к этому объекту. Причем для упрощения допустим, что имена объектов в системе

уникальны — имеется только одна бригада № 2, только один прессовый участок.

На этом завершим хождение «вокруг да около» и обратимся к примеру. В некоторой технологической системе контролируются выработка и время простоя нескольких одинаковых бригад, одинаковых цехов, одинаковых участков. Мы можем таким образом выделить три класса и поставить каждому в соответствие по одному процедурному файлу:

Наименование класса	Имя класса	Имя процедурного файла
бригады	BR	BR
цеха	CECH	CECH
участки	UTCH	UTCH

Очевидно, что поскольку это разные производственные подразделения, то структура баз, хранящих информацию о них, должна быть различной. Однако мы выделим общую «головную» базу HEAD.DBF со следующими структурой и содержанием:

OBJECTNAME (CHAR — 20)	CLASSNAME (CHAR — 8)	<другие поля, одинаковые для объектов всех классов>
бригада № 1	BR	
цех № 3	CECH	
бригада № 40	BR	
прессовый участок	UTCH	
кузнечный цех	CECH	
бригада Иванова	BR	
участок № 15	UTCH	

В программе определены только следующие действия для всех классов: расчет выработки и расчет времени простоя, — эти действия имеют одинаковое логическое назначение для всех классов, но различное внутреннее содержание для каждого класса.

В соответствии с изложенным создадим три процедурных файла, имена которых совпадают с именами классов: BR.PRG, CECH.PRG, UTCH.PRG. В каждом из этих файлов определены по две процедуры: calcwrk и timest, но реализации одноименных процедур в каждом из файлов различны. Рассматривать внутреннее содержание этих процедур мы не будем — предусмотрим, что именно внутри них устанавливаются необходимые связи между базами для объектов каждого класса, производится обращение к необходимым полям и выполняются соответствующие расчеты.

Подготовим теперь для примера несколько процедур, обеспечивающих полиморфный вызов методов объектов.

Процедура StartMtd1 (пример 1) предназначена для запуска необходимого одноименного метода всех объектов, отобранных в соответствии с каким-либо условием, и будет использоваться преимущественно внутри цикла scan...endscan, осуществляемого над базой HEAD.DBF.

Процедура StartMtd2 (пример 2) предназначена для запуска необходимого одноименного метода объекта, имя которого указано.

Пример 1

```
procedure StartMtd1
parameters Metod
do &Metod in &CLASSNAME
*запускается процедура с ранее  указанным  именем,
*находящаяся в процедурном файле, имя которого хранится
*в поле CLASSNAME
return
```

Пример 2

```
procedure StartMtd2
parameters Obj, Metod
locate for OBJECTNAME=Obj
if found()=.F.
 clear
 wait "Подразделение с этим именем не зарегистрировано!"
else
 do &Metod in &CLASSNAME
endif
return
```

Теперь подготовим небольшой управляющий экран для нашей системы (рис. 2).

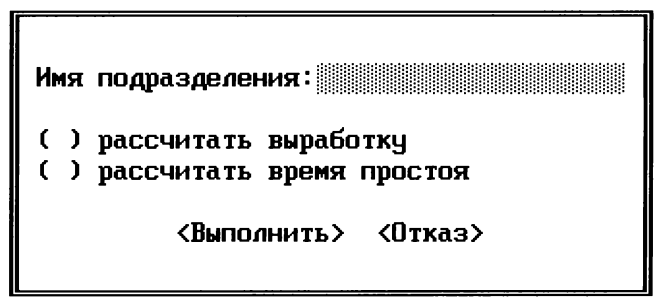


Рис. 2

Для экономии места рассмотрим только упрощенный текст valid-функции триггерной кнопки (пример 3).

Пример 3

```
function Trig_val
if Trigger=2
   clear read
else
   select HEAD
   MetodName="timest"
   if Metod=1
      MetodName="calcwrk"
   endif
*******************************************************
 if alltrim(Division)=""
   scan
      do StartMtd1 with MetodName
   endcsan
    *выполняется метод (расчет выработки или расчет
    *времени простоя) для всех зарегистрированных в базе
    *HEAD.DBF подразделений
 else
    do StartMtd2 with ObjName, MetodName
    *выполняется метод (расчет выработки или расчет
    *времени простоя) для подразделения с указанным
    *именем
 endif
endif
return .T.
*******************************************************
```

Работа функции заключается в следующем:

– если пользователь нашей системы ввел имя подразделения, то выполняется указанный в радиокнопке (переменная Metod) расчет только для этого подразделения, причем процедура StartMtd2 сама разбирается, к какому классу принадлежит указанное подразделение (или в каком процедурном файле находятся процедуры-методы для объектов этого класса) и вызывает процедуру с соответствующим внутренним содержанием;
– если пользователь не указывает имя подразделения (переменная Division изначально инициализируется пробелами), то выполняется указанный в радиокнопке расчет для всех подразделений, причем процедура StartMtd1 сама разбирается, к какому классу принадлежит очередное зарегистрированное в HEAD.DBF подразделение (или в каком процедурном файле находятся процедуры-методы для объектов этого класса) и вызывает процедуру с соответствующим внутренним содержанием.

Обратите внимание, какую большую работу выполняет фрагмент исходного текста функции, обрамленный звездочками, — в зависимости от ситуации он вызывает одну из шести процедур (нужную!), находящихся в трех процедурных файлах (подумайте, какую чехарду из do case...endcase вам пришлось бы написать при традиционном процедурном подходе к написанию этой системы, да еще рискуя запутаться в именах процедур, которые в этом случае были бы такими: calcwrk1, calcwrk2, calcwrk3, timest1, timest2, timest3).

Важным является также то, что при расширении функциональности системы (введении новых классов объектов, включении новых процедур-методов, независимо от их числа) вам не придется переписывать ни этот фрагмент, ни процедуры StartMtd1(2).

Предыдущий абзац проиллюстрируем двумя примерами:

– необходимо включить в систему подразделения (объекты) класса «СКЛАДЫ» (допустим, их несколько: склад № 1, склад № 2, основной склад), для которого определены те же методы. Вы создаете при необходимости дополнительные базы, в которых будет храниться специфичная для объектов этого класса информация и новый процедурный файл — SCKLAD.PRG, в котором пишете новые процедуры calcwrk и timest с соответствующим содержанием. В HEAD.DBF появляется несколько новых записей, поле CLASSNAME которых имеет значение «SCKLAD». В этом случае вам не придется изменить ни буквы из всего приведенного листинга, управляющий экран останется таким же;
– необходимо включить в систему новый метод «расчет выбраковки изделий» (calcbad) для объектов всех трех классов. В этом случае, возможно, вам придется добавить новые БД для хранения информации, необходимой в работе этого метода, и обязательно дополнить все три процедурных файла процедурой calcbad, имеющей три различных варианта внутреннего содержания. В этом случае радиокнопка управляющего экрана получит дополнительную опцию — «расчет выбраковки изделий». В связи с этим необходимо «обыграть» внутри valid-функции присвоение MetodName="calcbad". Больше ничего менять не нужно.

Отметим преимущества предлагаемого варианта по сравнению с традиционным.

1. Чисто техническое — явно уменьшился объем исходного текста, который отвечает за вызов нужных методов нужных объектов.

Следующие преимущества являются «идеологическими», то есть более существенными, чем

уменьшение объема текста, результатами применения новой технологии.

2. Каждому логически завершенному единому действию соответствует процедура, имеющая для разных классов одинаковое имя, но реализованная для каждого класса по-разному (это, в принципе, суть полиморфизма — возможность вызова по одному имени различных по содержанию процедур в зависимости от того, к какому классу принадлежит объект, для которого вызывается процедура).

   При традиционном программировании процедурам, выполняющим логически однородные действия для объектов различных классов, придется дать похожие, но все-таки разные имена, отражающие в своем названии то, что действие этих процедур относится к различным классам объектов.

3. Процедуры, реализованные для конкретного класса, выделены в отдельные процедурные файлы, что облегчает понимание функционирования системы и ее модификацию.

Иными словами, потратив 10-15 минут на программирование объектной оболочки, вы получили значительно более гибкую и легко модифицируемую конструкцию, чем при традиционном процедурном программировании.

Необходимо несколько слов сказать об основном «идеологическом» отличии изложенного здесь подхода от представленных в [1], [2].

Так как теперь принадлежность объекта к некоторому классу не выясняется в программном коде, а является атрибутом объекта, хранящимся в БД, то появилась возможность говорить о более глубокой реализации полиморфизма на уровне вызова методов объектов, а не методов классов, как предлагалось в упомянутых работах.

С чисто технической точки зрения, благодаря наличию в FoxPro v2.xx возможности вызова процедуры из любого файла, а не только подключенного командой set procedure to <имя файла>, вызов процедур-методов, вероятно, будет производиться быстрее, чем при использовании методики, описанной в [2]. По сравнению с [1] допустимые имена классов и методов стали длиннее (и содержательнее). ■

Литература

1. **А.Дубина, С.Сынгур. Имя класса + имя метода =... полиморфизм!// Монитор. — 1994. — № 8.**
2. **А.Дубина. Частичная эмуляция полиморфизма в программах на языке FoxPro// The Pinter FoxPro Letter (русское издание). — 1994. — № 2.**

НАДЕЖНАЯ СВЯЗЬ НЕ ВСЕГДА ДОРОГО

Модемы

Полный ряд модемов и факс-модемов от простейших до профессиональных и со скоростью до 28800 бит/с. Схемотехника и микропрограммы, разработанные ***HATEKC*** *совместно с американской фирмой-производителем, обеспечивают надежную связь на телефонных каналах СНГ.*

Пользовательские модемы ***UNICOM****, полностью адаптированные для отечественных сетей.*

Профессиональные модемы ***BULLET*** *с расширенным набором функций.*

Магистральные 2-х и 4-х проводные модемы ***TAICOM****, скорость от 14400 до 128000 бит/с.*

Поддержка систем голосовой почты и аппаратуры X.25 и X.400. Сертификат Министерства Связи России и ведущих телекоммуникационных сетей.

Терминалы

VERTEX, NVT - недорогие видеотерминалы для систем UNIX, IMS MDOS7, DOS Line, PC-MOS, PICK, VM/386, MUMPS, MSM ...

- *Эмуляция - VT52, VT100, VT220, PC TERM*
- *Монитор - Hercules, Mono VGA, Color VGA*
- *Кодировка - альтернативная, КОИ-8, ISO-8859*
- *Скорость обмена от 110 до 115200 бод.*
- *Интерфейсы - RS232C и Centronics*

Мультиплексоры

AST4, PCCOM8 *- наиболее популярные и* ***RS232C*** *недорогие 4-х и 8-ми канальные мультиплексоры. Позволяет устанавливать любой стандартный базовый адрес, переключатель инверсный/неинверсный регистр прерывания.*

HICOM 4, 8 *- скоростные варианты AST4 и PCCOM8 с внутренним буфером (****16C550UART****).*

ТЕЛЕКОММУНИКАЦИОННЫЕ СЕТИ

Поставка, разработка и производство под "ключ"

НТЦ "НАТЕКС"

тел. (095)324-4000, 325-1862;
факс (095)324-2807, 322-0879

Дилерам скидки до 40%

Нам пишут, мы отвечаем

А. Колесов

Предисловие к материалам рубрики

Сегодня мы публикуем письмо коллеги из Литвы, одного из типичных Basic программистов бывшего Союза. Отвечая на некоторые затронутые в письме проблемы, мы более подробно рассматриваем вопрос оптимизации программ на QuickBasic с помощью включения модулей NOEM.OBJ и SMALLERR.OBJ, входящих в состав дистрибутива QB 4.5.

Мы активно выпускаем наш электронный журнал «BasicArt» (к декабрю 1994 года подготовлено 4 таких сборника объемом 170-400 кб в архивированном виде). Кроме того, уже сейчас началась работа по распространению различных материалов по электронным каналам связи, в частности, по сети FIDOnet. Более подробную информацию по этому поводу можно получить по адресу:

105058 Москва, а/я 4, ИКЦ АПМБ,
А.А. Колесову, тел. (095) 366-25-02,
E-mail: akolesov@glas.apc.org

Уважаемый г-н Колесов!

С QuickBasic 4.5 я работаю уже второй год, а до этого два года работал с версией 4.0. Сейчас у меня компьютер AT\286, но в ближайшее время я надеюсь перейти на 386DX.

Профессиональным программистом себя не считаю, так как по образованию я физик и программирование для меня — лишь способ решения задач в своей предметной области. В этом плане система QB мне сильно помогла и, я бы не побоялся сказать, открыла новый взгляд на мир. Но это, конечно, вместе с появлением на столе PC. Мне приходилось решать различные задачи в области ультразвуковой техники: поиск резонансных частот в составных колебательных системах, определение параметров полей, излучаемых ультразвуковыми преобразователями систем медицинской диагностики. Мой опыт показывает, что QB позволяет достаточно оперативно воплотить модель в реальность и увидеть результат, что очень важно.

Значительное место в своих программах я уделяю пользовательскому интерфейсу, так как всегда хочется, чтобы с твоими программами было удобно работать другим людям. Причем каждый раз приходится изобретать что-то новое, но в итоге все сводится к уже принятым формам в виде диалоговых окон, панелей меню и так далее. Своеобразную универсальность подхода к решению подобных проблем можно увидеть в системе Windows и Windows программах. Именно с этой точки зрения мне интересен Visual Basic, поскольку появляется надежда облегчить свой труд в области интерфейса и одновременно повысить эстетическое качество программы. Опыта работы с VB у меня нет, но знаком с ним по книгам К. Брауна и Х.-Г. Райсманса в переводе на русский язык.

Несколько слов о своих проблемах. К сожалению, пакет QB 4.5 у меня неполный, не хватает файлов MOUSE.COM, BRUN45.EXE, демонстрационных программ. Попытки найти их были тщетными. Среди программистов почему-то бытует мнение, что Basic — это несерьезно. Поэтому очень важным является общение и связь между пользователями языка.

Интересно было бы узнать также, как с помощью LINK.EXE связать объектный файл с NOEM.OBJ. Мои попытки это сделать не привели к успеху — компоновщик неизменно выдавал сообщение, что не находит NOEM.OBJ, хотя находился в одном каталоге с ним.

С уважением, Сергей Крючков
Литовская Республика, Вильнюс

Уважаемый г-н Крючков!

Прежде всего, мне приятно констатировать, что появление новых государственных границ не является препятствием для продолжения нормальных контактов между жителями независимых государств.

Описываемая в вашем письме ситуация с программированием на Basic является достаточно характерной практически для всех регионов бывшего Союза. Именно пытаясь как-то решить проблему информационного вакуума и обеспечения контактов между программистами, группа энтузиастов Basic создала в прошлом году информационно-консультационный центр «Ассоциация пользователей Microsoft Basic». Более подробную информацию о его работе вы можете получить в журнале «Монитор» 3.94. В середине прошлого года мы также договорились с редакцией журнала о создании постоянной рубрики «MS BASIC».

К сожалению, приобретение пакета QB 4.5 действительно является проблемой для большинства Basic программистов. И дело здесь не только в том, что он стоит 99$, главное — его уже нет в российских каталогах «Microsoft», хотя в Америке и Европе он еще кое-где продается. То же самое относится и к пакету PDS 7.1: он продается, но фактически в Россию не завозится — считается неходовым товаром.

Ваше замечание об отсутствии файла MOUSE.COM я понял как объяснение невозможности работы с «мышью» в прикладных программах. Но ведь MOUSE.COM — это просто драйвер «мыши», который «Microsoft» включает во все свои продукты (вполне возможно, что для вашего компьютера он и не подойдет — всегда лучше использовать тот драйвер, который поставляется на дискете вместе с самой «мышью»). А проблема в работе с «мышью» в прикладной программе совсем в другом — надо уметь правильно работать с соответствующим прерыванием MS-DOS. Этот вопрос уже неоднократно описан в компьютерной прессе, но, наверное, имеет смысл рассмотреть его применительно к Basic в нашей рубрике в ближайшее время.

Типичной ситуацией являются и ваши проблемы с пользовательским интерфейсом, и интерес в связи с этим к Visual Basic. Тема «Visual Basic» уже кратко затрагивалась в нашей рубрике и будет расширяться в дальнейшем. Но тем не менее, не сомневаясь в перспективах этой системы программирования, мне хотелось бы заметить, что и для систем Quick и PDS существует огромное число зарубежных и некоторое количество отечественных вспомогательных средств, библиотек самого различного назначения: интерфейс, графика, шрифты, работа с базами данных и прочее. Эти средства способны реально помочь Вам уже сегодня, учитывая то, что даже мощности компьютера 386DX может оказаться недостаточно для VB/Win 3.0.

В расчетных задачах не забывайте о FORTRAN

Что касается вопроса оптимального выбора языков программирования, то с учетом специфики задач нашего читателя (главная цель — математические расчеты) я бы посоветовал подумать о разделении проблемы на две части:

а) интерфейс и визуализация

и

б) выполнение математических расчетов.

Причем для второй части, возможно, гораздо удобнее и эффективнее использовать старый добрый FORTRAN — именно на его основе продолжает создаваться сейчас большинство расчетных систем в США. Во-первых, это обеспечивает реальную гибкость при переходе с одной платформы на другую (у нас в лаборатории и сейчас работают без переделки расчетные программы, написанные для ЕС ЭВМ лет 10-15 назад), а во-вторых, новая 32-разрядная версия Microsoft FORTRAN Power Station 1.0 уже сегодня позволяет реально и в полной мере использовать возможности архитектуры 32-разрядных компьютеров и расширенной памяти (разговор о создании 32-разрядного Basic пока ведется только в плане неясных перспектив) создавать программы MS-DOS и для Windows.

Об использовании модуля NOEM.OBJ

Из вашего письма не совсем ясно, в чем конкретно заключалась проблема с подключением модуля. Для большей ясности в следующий раз присылайте более точное описание всех выполняемых операций (командных строк DOS, ввода дополнительных параметров) и соответствующих диагностических сообщений системных программ. В данном случае складывается впечатление, что вы просто неправильно обратились к LINK.EXE (работа вне среды требует определенного опыта). Поэтому я попробую описать последовательность решения данной задачи с самого начала.

В составе дистрибутива QuickBasic 4.5 есть два объектных файла, позволяющих оптимизировать получаемый EXE-модуль. Это NOEM.OBJ (NO EMulation), который обеспечивает прямую работу с сопроцессором (80×87), блокируя использование процедур эмуляции его операций, а также SMALLERR.OBJ, отменяющий вывод сообщений об ошибках при выполнении EXE-модуля. Соответственно их использование сокращает размеры исполняемого модуля, а применение NOEM.OBJ — еще и время выполнения программы.

Как известно, существует *три способа* выполнения операций с плавающей запятой в программах. *Первый* использует прямое обращение к математическому сопроцессору и подразумевает его обязательное наличие. Сообщение при запуске программы типа «Floating point error» говорит, что программа использует именно этот вариант, а сопроцессора на компьютере нет. Здесь надо заметить, что возможна такая ситуация, когда сопроцессор физически установлен на материнской плате, CHECKIT его показывает, но в SETUP вы забыли установить его — обращения к нему не доходят.

Второй способ использует процедуры альтернативной математики, реализованные на уровне библиотек подпрограмм. Он обходится без сопроцессора и работает для операций с плавающей запятой существенно медленнее первого.

Третий вариант является комбинацией первых двух и называется режимом эмуляции. Здесь сначала производится проверка наличия сопроцессора, а затем выполняется обращение либо к нему, либо к процедурам альтернативной математики. Очевидно, что программа в этом случае получается самой большой по размерам и работает немного медленнее 1-го варианта на компьютере с сопроцессором и 2-го варианта без него. Но, в отличие от 1-го варианта, может работать без сопроцессора и работает существенно быстрее с сопроцессором по сравнению со 2-м.

Короче говоря, первые два варианта используются только, когда вам точно известна конфигурация компьютера и вы хотите повысить быстродействие и размеры программы (я думаю, в пределах 5-10%).

В системе QB 4.5 по умолчанию применяется именно вариант с эмуляцией, функции которой выполняет модуль PIEMR. Чтобы использовать модуль NOEM.OBJ для прямого обращения к сопроцессору, проще всего откорректировать системную объектную библиотеку BCOM45.LIB следующим образом (только предварительно сохраните старый вариант библиотеки):

```
LIB.EXE BCOM45.LIB-PIEMR+NOEM;
```

Следует отметить, что модуль NOEM заменяет только 3 входные точки из четырех, которые имеются в модуле PIEMR, но на одну точку — __fpemulator, кажется, никаких внешних ссылок в библиотеке нет.

Размер модуля PIEMR.OBJ — 9489 байтов, NOEM.OBJ — 1841, соответственно размеры библиотеки BCOM45.LIB — 220919 и 209143, а величина моего довольно большого EXE-модуля (работа со специальной базой данных) BASECOL.EXE — 146976 и 139404. К сожалению, мы имеем дело в основном с системами, использующими целочисленную арифметику, для которых эффект такой замены, с точки зрения быстродействия, весьма незначителен. Поэтому, если наши читатели, работающие с расчетными задачами, сообщат нам о полученном эффекте по быстродействию, то это будет очень интересно.

Но здесь нужно уточнить, что все сказанное относится только к EXE-модулям, созданным в режиме Stand-Alone. Дело в том, что в режиме Requiring... все операции эмуляции «зашиты» в модуле поддержки BRUN45.EXE, который нельзя откорректировать.

Точно так же можно подключить (только для режима Stand-Alone!) и модуль SMALLERR.OBJ, при этом из исходной библиотеки надо убрать два объектных модуля:

```
LIB.EXE BCOM45.LIB-MESSAGES-ERFETCH+SMALLERR;
```

Здесь эффект от сокращения размеров менее заметен: BCOM45.LIB — 207601 (с модулем NOEM), BASECOL.EXE — 138126.

В заключение надо сказать, что в системе Basic PDS 7.1 и VB/DOS есть несколько опций оптимизации EXE-модуля, которыми можно управлять непосредственно из среды QBX и которые действуют для обоих режимов создания исполняемого модуля — Stand-Alone и Requiring.... В частности, при установке пакета на жесткий диск SETUP предлагает сделать сразу два варианта библиотек и модулей поддержки — для режима эмуляции и режима альтернативной математики. И здесь надо обратить внимание на один тонкий момент.

При создании EXE-модуля вы можете использовать опцию /FPi (80x87 or Emulator Math), которая работает с библиотекой BCL71EFR.LIB. Но в данном случае по имени библиотеки нельзя узнать реализованный в ней режим работы с операциями плавающей арифметики, это зависит от выбранных вами при установке пакета опций. Дело в том, что при создании данной библиотеки вы можете указать возможность «обязательное наличие сопроцессора», в результате чего получите вариант работы с прямым обращением к сопроцессору.

Кроме того, при использовании модуля поддержки BRT71EFR.EXE нужно следить за его соответствием библиотеке BRT71EFR.LIB, использованной при создании исполняемого модуля. В противном случае при попытке запуска программы будет выдано сообщение об ошибке. Неприятным следствием этого является то, что в один программный комплекс нельзя включать EXE-модули, созданные с разными вариантами библиотеки BRT71EFR.LIB, — они должны работать с одним вариантом модуля поддержки.

Андрей Колесов

Журнал «Монитор» можно приобрести в городах России, Украины, Белоруссии, Узбекистана, обратившись к нашим официальным дилерам.

1. Баку
Фирма «Марко», ул.Хагани, 33, тел. (8922) 98-90-82.

2. Владивосток
Фирма «Компьютерные технологии», тел. (4232) 221-291.

3. Волгоград
Арендное предприятие «Техническая книга», тел. (8442) 363-597.

4. Днепропетровск
Попов А.Е., тел. (0562) 428-593.

5. Железноводск
Фирма «Кросс, Лтд.», тел. (6533) 99-563, 94-468.

6. Иркутск
Мадьяров Игорь Иосифович, тел. (3952) 46-19-68

7. Киев
Фирма «Диалектика», тел. (044) 266-40-74;
Фирма «ОКС», тел. (044) 446-95-05.

8. Минск
НПП «Триумф», тел. (0172) 661-609, 661-680, факс 666-606.

9. Пермь
Попов Андрей Леонидович, тел. (3422) 33-30-91

10. Ростов-на-Дону
Дмитрий Лагунов, тел. (8632) 24-13-10.

11. Санкт-Петербург
Дом книги (812) 219-94-57;
Магазин «Техническая книга», тел. (812) 164-65-65.

12. Тамбов
АО «От Елены» ЛПС «Shift», тел. (0752) 22-44-63.

13. Ташкент
Центр молодежных исследований, тел. (3712) 688-847.

14. Уфа
ПКФ «Шамиль», тел. (3472) 23-79-78.

15. Харьков
ЧНПФ «Харьков», тел. (0572) 23-37-22, факс 44-10-00.

16. Чебоксары
Фирма «Демос», тел. (8352) 208-395, 426-492, факс 209-437.

Приглашаем дилеров, тел. (095) 238-99-11.

ЭЛЕКТРОННЫЕ КЛЮЧИ

SOFTWARE SECURITY, INC. (США)

- Уникальный код в каждом ключе
- Встроенная энергонезависимая память
- Подключение к параллельному порту
- Прозрачность для принтера

AEGIS — обеспечивает надежную защиту готовых программ в среде DOS/WINDOWS. Позволяет шифровать файлы данных и внешних оверлеев.

ACTIVATOR — инструмент для реализации собственных идей по защите программ. Дополнительно поставляются библиотеки объектных процедур **OBJECT MODULES и TOOLBOX.**

DEMOKIT — демонстрационный комплект программно-аппаратных средств позволяет испытать все наши комплексы.

Более десяти лет во всем мире и уже три года в России и СНГ мы помогаем защищать авторские права разработчиков программного обеспечения.

SOFTWARE SECURITY BELARUS
Республика Беларусь, 220026
г. Минск, а/я 12
Тел./факс: (0172)45-21-03,45-31-61,
e-mail: lev@ssb.ibibel.glas.apc.org

Организация псевдомногозадачности в Turbo Vision

С. Миллер

Статья направлена на решение проблемы организации многозадачного режима работы (фоновых процессов) в Turbo Vision. Она является логическим продолжением статьи О. Бунина «Организация фоновых процессов в Turbo Vision».

В процессе программирования на Turbo Pascal (Borland Pascal) с использованием объектно-ориентированной библиотеки Turbo Vision многие программисты сталкиваются с проблемой организации многозадачного режима работы программ. Необходимость в этом появляется при выполнении длительных процессов, таких как вывод на принтер, расчеты с использованием большого количества данных и других. Многие пытаются применять метод Idle объекта TApplication, который вызывается безусловно во все неиспользуемые временные окна работы программы. Однако он неудобен: приходиться дробить процедуру на множество элементарных участков и вводить программный выбор нужного участка. В результате такого подхода часто появляется малопонятный и громоздкий код. Если к этому добавить желание иметь множество одновременно исполняемых процедур, особенно разнородных, то задача становится практически неразрешимой. Одно из решений данной проблемы было предложено О. Буниным в статье «Организация фоновых процессов в Turbo Vision», опубликованой в журнале «Монитор» № 3 за 1994 год. Несмотря на оригинальность решений и большую пользу описываемой библиотеки фоновых процессов, имеется ряд крупных недостатков, сужающих область ее использования. Бунин сам отмечает три ограничения.

1. В каждый момент времени может выполняться только один фоновый процесс.
2. В процессе исполнения фоновой процедуры следует открывать только немодальные окна и работать с ними, так как открытие модального окна «замораживает» фоновый процесс до тех пор, пока оно не будет закрыто.
3. Не должно быть ситуаций, когда фоновый и основной процессы одновременно обращаются к магнитному диску или используют иные нереентерабельные функции Pascal и самого приложения: New, GetMem и тому подобные.

Третий пункт, по моему мнению, нельзя назвать ограничением, так как его легко обойти, но к оставшимся двум хочется добавить следующее.

Модуль Fon, описанный в статье Бунина, использует модуль TPINT из библиотеки Turbo Professional версии 5.05 или 5.09, в которой применяются всего четыре функции, но необходимо приобретение всего пакета. Пакет Turbo Professional не является объектно-ориентированным, и в программе с использованием Turbo Vision TPINT является чужеродным.

Модуль организации фонового процесса Fon написан не по правилам объектно-ориентированного программирования. За счет этого теряется гибкость, а главное, ломается общая концепция построения программы в

Turbo Vision. Turbo Pascal (Borland Pascal) допускает такую вольность, но назвать это хорошим стилем нельзя.

Данная статья направлена на решение проблемы организации многозадачного режима работы (фоновых процессов) в Turbo Vision. Являясь логическим продолжением статьи О. Бунина «Организация фоновых процессов в Turbo Vision», она решает практически все проблемы и снимает ограничения, присущие модулю Fon из статьи О. Бунина.

Библиотека многозадачного режима в Turbo Vision состоит из одного TPU-файла. В качестве исходных модулей для него служат MultTask.pas и Switch.asm, который подключается директивой {$L ...} и содержит низкоуровневые процедуры переключения задач. Модуль MultTask.pas транслируется Turbo Pascal 6.0 или Turbo Pascal 7.0 (Borland Pascal 7.0) — turbo.exe, tc.exe, bp.exe или bpc.exe. Модуль Switch.asm транслируется любым Turbo Assembler, например tasm.exe, который входит в комплект поставки Turbo Pascal. Отметим особенность ассемблерного модуля — в нем не задана модель трансляции. Это сделано с целью избежать появления дополнительного кода при трансляции. Некоторые процедуры завершаются нестандартно, например командой IRET, что может вызвать неправильную трансляцию модуля при указании модели.

ПРОГРАММЫ

Основой псевдомногозадачной среды является объект TMultiTaskApp — потомок TApplication. Он наследует все свойства родителя, кроме того, добавлены функции управления процессами.

Основой многозадачной среды является объект TMultyTaskApp — потомок TApplication. Он наследует все свойства родителя и, кроме того, имеет свои поля и методы для реализации примитивов управления задачами. Он может осуществить функции добавления, переключения, временного замораживания и удаления задач, созданных головным процессом (собственно ядром Turbo Vision). Напрямую TMultyTaskApp едва ли будет использоваться в пользовательской программе. Создавая потомков этого объекта, пользователь получает готовый механизм управления задачами через поля и методы родителя. Множество разнородных задач хранится в виде коллекции процессов TTask, благодаря полиморфизму TCollection. Добавление очередной задачи (фонового процесса) происходит следующим образом:

– в динамической области памяти (heap) создается стек задачи;
– конструктором TTask в стеке заполняется начальный дескриптор;
– задача включается в коллекцию — Tasks методом NewTask.

В случае использования динамической области памяти для размещения объектов простейший пример включения очередной задачи следующий:

```
TMultyTaskApp.NewTask(New(PTask, Init));
```

В объекте TTask основным исполняемым методом является Executer, он пустой, виртуальный и может перекрываться. Бывают случаи, когда на время нужно заморозить выполнение определенного процесса или всех процессов (задач) в системе. Для этого необходимы методы объекта TmultyTaskApp:

* для замораживания одного процесса или всех сразу используются FreezeTask и FreezeAll соответственно;
* для размораживания применяются методы UnFreezeTask и UnFreezeAll, для удаления одной или всех задач из коллекции TermTask и TermAll соответственно. Дополнительно объект TMultyTaskApp имеет некоторые сервисные методы:
* PreTaskSw вызывается при порождении первого процесса (задачи) — первого элемента коллекции Tasks;
* PostTaskSw вызывается после удаления последней задачи из коллекции Tasks.

Эти методы являются пустыми, но перекрываются пользователем для выполнения некоторых действий в своей программе, например для разрешения или запрета ряда команд в Turbo Vision. Для обработки ошибок создания и размещения задач существует виртуальный метод TaskError. Объект TTask служит основой для создания пользовательских задач-потомков. В свою очередь TTask является потомком от TObject, что дает возможность его простого использования в коллекции — TCollection. Поля этого объекта содержат:

* Stack — указатель на стек задачи, находящийся в динамической области и захватываемый в конструкторе TTask.Init. Размер стека определяется в глобальной переменной (типизированной константе) TaskStackSize. Ее значение может изменяться пользователем до вызова конструктора объекта для изменения величины стека задачи. Начальное значение равно 32 килобайтам, что достаточно практически для любого процесса. Для небольших фоновых задач значение этой переменной необходимо снижать в целях эко-

номии памяти. Указатель Stack доступен только по чтению.

* TaskSSSP — указатель, равный текущему значению SS:SP задачи. Остальные значения регистров находятся в локальном стеке-дескрипторе задачи, на который указывает поле Stack. При переходе на задачу все регистры ЦПУ выбираются ассемблерными командами pop XX, где XX обозначает все регистры общего назначения процессора 80x86. Поле TaskSSSP доступно только по чтению.
* Поле Freeze определяет состояние процесса: заморожен, если равно true, в противном случае false. Это поле модифицируется из основного объекта TMultyTaskApp.
* Поле NTacts является служебным. Оно содержит текущее значение счетчика тиков (только чтение), которое служит для вычисления приоритета процесса.

  Примечание. Тиком будем называть событие от таймера.
* Поле Ticks содержит текущий приоритет процесса. По сути дела оно определяет, после какого запроса на переключение произойдет переход на выполнение пользовательской задачи. Чем больше значение этой переменной, тем реже будет происходить обращение. При инициализации в конструкторе объекта TTask ей присваивается значение 5. Изменить приоритет можно, вызвав метод TTask.SetPriority, где в качестве параметра передается новое значение приоритета.

Для выполнения нереентерабельных функций Pascal или DOS в пользовательской задаче применяется механизм монопольного исполнения критического участка: перед входом в него блокируется механизм переключения задач. После прохождения критического участка фоновая задача должна разблокировать переключение задач. Для блокировки вызывается метод LockTV, а для разблокирования UnLockTV объекта TTask. Эти методы воздействуют на содержимое глобальной переменной TVAvailable, которая отвечает за разрешение переключения с пользовательской задачи на основной процесс. Попутно отметим, что выполнение критических участков основного процесса алгоритмически выполняется в монопольном режиме.

Рассмотрим технологию создания новой пользовательской задачи и процесс переключения задач в системе. Как было отмечено, новая задача создается пользователем как потомок от TTask. Для созидательной деятельности пользователь перекрывает виртуальный метод Executer, который собственно и является исполняемым телом задачи. При вызове конструктора TTask.Init происходит захват памяти под стек, затем значение указателя на стек смещается в самый верх захваченной памяти. Для корректногоосвобожденияпамятисохраняетсяреальный адрес захвата и размер стека. Этот алгоритм позаимствован из модуля TPINT библиотеки Turbo Professional. После того как подготовлен стек, происходит вызов процедуры InitTaskStack из ассемблерного модуля Switch.asm. Эта процедура заполняет в стеке дескриптор задачи:

* начальные значения регистров (вплоть до CS:PC);
* указатель на @Self объекта TTask; это делает Turbo Pascal при вызове любого метода объекта;
* указатель на процедуру завершения задачи — EndTask. После выполнения команды RET метода Executer произойдет переход на этоту метку. В результате завершенная задача будет автоматически удалена из динамической области памяти и из коллекции задач.

После заполнения стека процедура InitTaskStack возвратит значение SS:SP задачи. Оно помещается в поле текущего значения SS:SP задачи — TaskSSSP.

После выполнения действий конструктора TTask.Init (и плюс, если нужно, пользовательских инструкций, связанных с инициализацией объекта, порожденного от TTask) вызывается метод TMultyTaskApp.NewTask. Как было отмечено, этот метод вставляет в коллекцию Tasks очередной указатель на задачу. В случае неудачной инициализации задачи вызывается виртуальный метод TMultyTaskApp.TaskError. Он является пустым, но может быть перекрыт пользователем для реакции на ошибку размещения задачи в динамической области памяти. На этом процесс создания новой задачи завершается.

Рассмотрим механизм переключения задач в системе. Для этого был выбран алгоритм переключения по таймеру IBM PC, использующий захват пользовательского вектора прерывания от таймера — $1C. По таймеру происходит только выход из фонового процесса (задачи). Переход на задачу осуществляется из метода Idle объекта TMultyTaskApp. Такой подход дает возможности пользователю не учитывать вхождение в нереентерабельные функции Pascal и Turbo Vision основного процесса. Это связано с тем, что метод Idle вызывается только в свободные кванты времени, то есть не происходит прерывания процедур и функций основного процесса. Так как метод Idle вызывается регулярно, нет задержек с переключением задач, в отличие от библиотеки Бунина, в которой была предпринята попытка

приготовить жаркое, вращая вокруг него очаг, то есть вызывать ядро Turbo Vision из фоновой задачи. После передачи управления методу TMultyTaskApp.Idle происходит проверка условий для вызова методов PreTaskSw и PostTaskSw (листинг 1). Включение этой проверки в метод Idle сделано для безусловного перехода на PreTaskSw и PostTaskSw независимо от способов порождения и уничтожения задач в системе. После этого подыскивается подходящая задача для переключения.

Алгоритм поиска следующий: последовательно перебирают все задачи из коллекции Tasks и ищут ту, у которой совпали значения приоритета Ticks со значением текущего значения счетчика тиков задачи — NTacts. После сравнения счетчик тиков инкрементируется. В случае, если хотя бы одна задача удовлетворила условию Ticks = NTacts, выполняется следующее:

* вначале запоминается ее номер для равномерного вызова задач;
* происходит переключение на нее при помощи функции GoToTask.

Функция перехода на задачу GoToTask возвращает признак завершения задачи true или признак прерывания задачи по таймеру false, по которому происходит исключение задачи из коллекции TMultyTaskApp.Tasks. Кроме того, GoToTask модифицирует значения SS:SP текущей задачи (TTask.TaskSSSP), а также дескриптор задачи. Рассмотрим самый тонкий и ответственный участок — собственно переключатель задач. Низкоуровневые процедуры находятся в модуле Switch.asm. Это — GoToTask, ReturnTask и EndTask, а также процедура обработки прерывания по таймеру ClockInt (листинг 2). После принятия решения о переключении на задачу управление передается на функцию GoToTask, которая выполняет следующие действия:

– сохраняет некоторые регистры, необходимые Turbo Pascal и самой функции переключения в стеке основного процесса;
– сохраняет значения SS:SP основного процесса;
– устанавливает новое значение регистровой пары SS:SP центрального процессора 80x86;
– выбирает все регистры CPU 80x86 из локального стека задачи, последними выбираются CS:PC и регистр флагов командой IRET.

Таким образом происходит переход на приостановленное место в методе Executer объекта, порожденного от TTask, как будто после прерывания. По истечении одного тика (18 миллисекунд) вызовется прерывание по таймеру. После выполнения стандартного обработчика прерывания таймера управление передается на процедуру обработки прерывания ClockInt. В случае если не был блокирован выход из фоновой задачи методом TTask.LockTV, процедура ClockInt вернет управление в метод Idle. При этом будет установлен признак прерванного процесса — False, который передается как возвращаемое значение функции GoToTask. Перед выходом процедура ClockInt посылает контроллеру прерываний команду окончания аппаратного прерывания (в порт $20 заслать $20). В случае если метод Executer выполнился до конца, управление передается на метку EndTask модуля Switch.asm. Адрес метки EndTask заботливо занесла в стек задачи функция InitTaskStack при инициализации задачи. После перехода на EndTask выполняются все действия, как и по прерыванию задачи, за исключением того, что в метод Idle возвращается признак завершения задачи — True.

Для демонстрации основных возможностей библиотеки псевдомногозадачной среды в Turbo Vision была создана программа New.pas (листинг 3). Она транслируется в исполняемый файл New.exe и включает, кроме стандартных модулей (*.TPU) Turbo Vision, модуль MultTask.pas (.TPU). Основой программы служит объект TFonApp — потомок TMultyTaskApp, в котором перекрыты следующие методы.

– Init — для уменьшения размера стека пользовательских задач до 1024 байтов. Для этого изменяется значение глобальной переменной TaskStackSize из модуля MultTask.pas. Также запрещаются команды работы с задачами, так как при запуске программы задачи отсутствуют.
– InitMenuBar — для добавления команд работы с задачами, также включает стандартный набор действий с окнами.
– HandleEvent — для выполнения команд по управлению задачами.
– PreTaskSw — для разрешения команд работы с задачами при порождении первой задачи.
– PostTaskSw — для запрещения команд работы с задачами при удалении последней задачи. Отметим, что последние два метода должны содержать короткий и быстро выполняемый код, как и все Idle-процессы.

Также в объект TFonApp добавлены методы для создания и включения в систему новой задачи — OpenTask, для удаления задачи — TermTask, для временного замораживания всех задач — FreezeTask и их размораживания — UnFreezeTask. После запроса пользователя на создание очередной задачи путем выбора из меню или по «горячей» клавише происходит генерация команды cmNewTask, создается задача, копия объекта TDemoTask — потомка от TTask. В этот объект до-

бавлено поле PW — указатель на TDemoWindow, потомок от видимого объекта Turbo Vision — TWindow, который служит для отображения последствий «бурной деятельности» перекрытого метода Executer. Действия этого метода заключаются в периодическом изменении символа заполнения окна — TDemoWindow. В конструкторе объекта TDemoTask — Init происходит инициализация TDemoWindow, ему передается указатель на @Self (на себя), после чего TWindow вставляется в DeskTop объекта TMultyTaskApp процедурой InsertWindow. Процедура InsertWindow добавлена в Turbo Vision 2.0, она производит проверку на корректность вставляемого окна перед вызовом метода DeskTop^.Insert. Здесь надо отметить тонкость такой операции — вставляемое окно имеет двух владельцев: во-первых, это копия объекта TDemoTask, а во-вторых, DeskTop копии объекта TFonApp. Двойное владение влечет за собой опасность потери указателя на объект одним владельцем при удалении его вторым владельцем. Например, если попытаться вставить в метод TDemoTask^.Executer команду закрытия окна, то в момент ее выполнения может произойти «зависание» системы, если пользователь изменял размер или перемещал окна, то есть Turbo Vision работал с указателем на это окно. Причина «зависания» в том, что ситуация «выдергивания окна из-под рук» Turbo Vision не отрабатывается, он продолжает работать с уже несуществующим указателем. Необходимо отметить, что в TDemoWindow перекрыт метод HandleEvent для отслеживания реакции на закрытие окна пользователем. В случае поступления команды cmClose не только закрывается окно, но и удаляется фоновая задача. Пример такого алгоритма встречается, например, при распечатывании файлов в системе Windows, когда при нажатии на клавишу Cancel на панели Printing происходит прекращение печати с удалением этого диалогового окна с экрана.

Внимание! Для правильной работы псевдомногозадачной библиотеки при использовании DOS 5.0 или выше необходимо в файл CONFIG.SYS включить строку STACKS=0,0. Это ограничение связано с тем, что DOS создает свои локальные стеки для аппаратных прерываний, а так как в библиотеке используется нестандартный выход из прерывания по таймеру, то переполняется внутренний стек DOS. К тому же при установке внутреннего DOS-стека нулевой длины дополнительно освобождается память в нижних 640 килобайтах ОЗУ. Заметим, что при отсутствии команды STACKS=0,0 DOS по умолчанию устанавливает значение STACKS=9,128, что соответствует созданию девяти внутренних стеков по 128 байтов каждый. Для снятия данного ограничения необходимо изменить алгоритм обработки прерывания по таймеру следующим образом.

- В модуле MultTask.pas при инициализации захватывать не пользовательское прерывание по таймеру $1C, а непосредственно прерывания по таймеру $08.
- Переписать обработчик прерывания ClockInt из модуля Switch.asm. В начале обработки прерывания необходимо вызвать старый DOS-обработчик таймерного прерывания. Для этого необходимо эмулировать команду INT:

```
pushf
call dword ptr [Old1C]   ; ds должен содержать seg DATA
```

Кроме того, из обработчика прерывания необходимо исключить команду окончания аппаратного прерывания (mov al,20h; out 20h,al), так как эта последовательность выполняется в DOS-обработчике прерывания.

- При выходе из программы восстанавливать старое значение вектора $08, а не $1C в модуле MultTask.pas.

Данную псевдомногозадачную библиотеку можно использовать также для создания задач, которые используют один или множество нераспределяемых ресурсов компьютера, например принтер. Для этого достаточно завести массив или коллекцию ресурсов, где будут отражаться состояния ресурсов, и в случае захвата ресурса какой-либо задачей все остальные задачи стоят в коротком цикле, ожидая его освобождения. Построение такого рода семафора распределения ресурсов является простейшим вариантом для данного класса задач. Этот алгоритм не учитывает приоритета распределения ресурсов. Также он предполагает, что в каждой задаче используется только один ресурс. В более сложных системах, применяющих различные законы распределения ресурсов, для управления ими, исключения конфликтов и тупиков существует целая теория управления процессами, описание которой заняло бы слишком много времени, но основа любой сложной параллельной системы, которая помещается в оперативную память, заложена в данной библиотеке.

Хочется надеяться, что эта библиотека поможет многим программистам создавать удобные и конкурентоспособные программы, которые всегда готовы отреагировать на любой запрос пользователя. ■

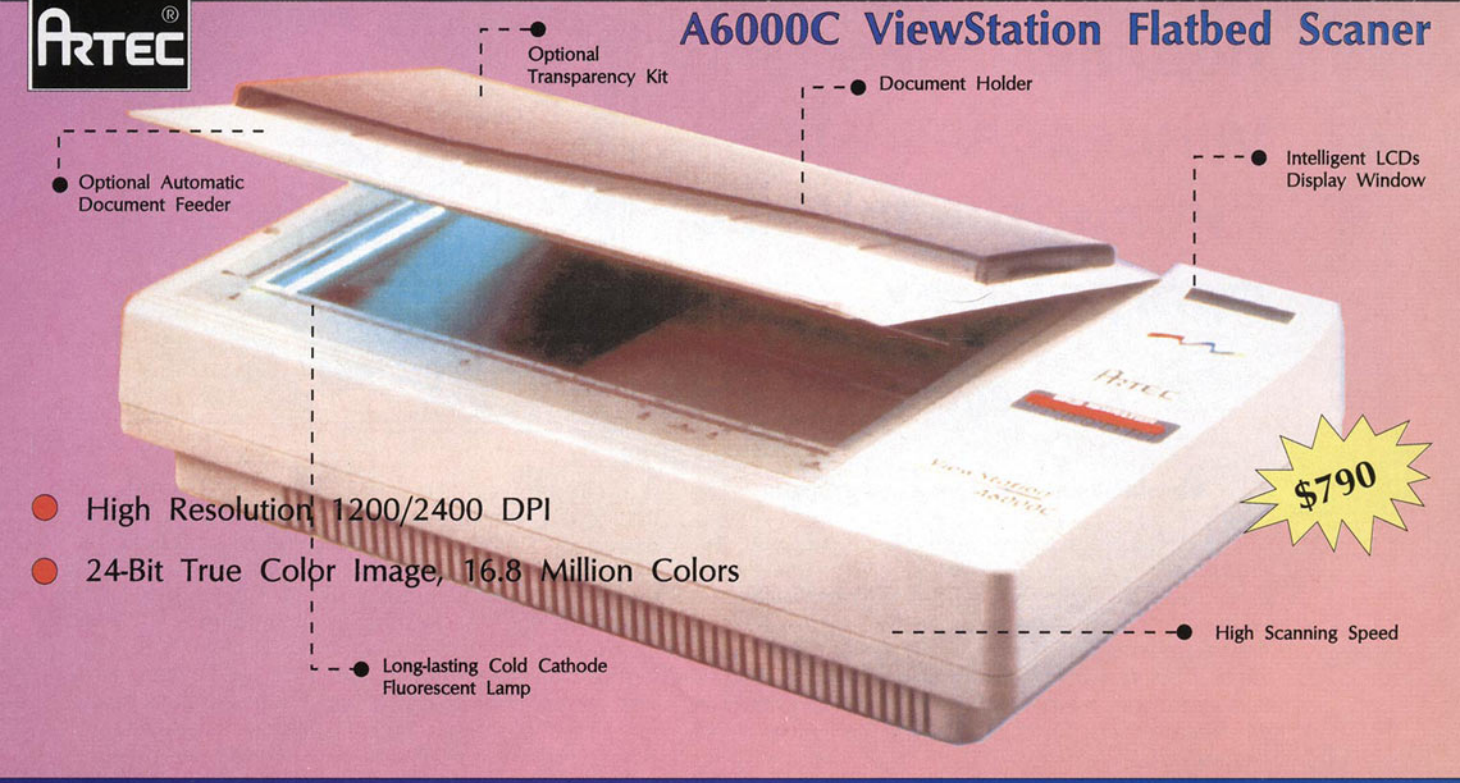
Trans-Ameritech
Enterprises, Inc.

High Performance
Computer Systems

Authorized Distributor

ARTEC®

A6000C ViewStation Flatbed Scaner



ViewScan/256

- 100-800 DPI, B/W & 256 True GrayScalel
- Magic Auto Stich/Merge Function
- PhotoStacker Plus & GO-CR Software

WalkScan/256

- 100-800 DPI
- Portable Parallel Port
- AC-adapter

ViewScan/COLOR 16M

- 100-1600 DPI
- 24-Bit Color Images, 16.8 Million Colors
- PhotoStacker Pro & GO-CR Software

BEST OF Computex BYTE

IMAGING MAGAZINE EDITORS' CHOICE

MIT

Dealers and Resellers WANTED CHP

TWAIN COMPATIBLE

Наши дилеры:
Сочи
ТОО "АРГО"
(8622) 93-6220

Московский Офис
119361 Москва
Большая Очаковская, 14

Тел. / факс : (095)
437-9745, 437-9746
430-2457, 430-9959

Наши дилеры:
Хабаровск
СП "СТЕЛЛАР"
(4212) 34-1679

Иваново	Тамбов	Екатеринбург	Челябинск	Белгород	Тула	Курган	Вятка
АО "РУСЛАН"	АО "СЕРВЕР"	ТОО "АРКОН"	АО "ПРОМАТИС"	ТОО "ЭЛТОН-Л"	"ЧЕРМЕТИНФОРМ"	ТОО "ФЕРУС"	фирма "КОРСА"
(0932) 23-3482	(0752) 33-5005	(3432) 44-1660	(3512) 65-5620	(07222) 71-552	(0872) 43-6068	(35222) 64-896	(8332) 67-5588

Листинг 1

```
{          Модуль для создания
         псевдомногозадачной среды
           (фоновых процессов)
             в Turbo Vision

       для транслятора Turbo Pascal
            и Borland Pascal
  для реального режима работы процессора 80x86

  Протестировано:
    CPU : 80286, 80386DX, 80486DX
    RAM : 1Mb, 4 Mb
    OS  : DOS 5.0
    TV  : Turbo Vision v2.0
    PAS : Borland Pascal v7.0

  Разработано:
    Миллер Станислав Юрьевич
    Москва / 1994
}

{$F+}
Unit MultTask;
                              Interface
uses
  Objects, App;

type
  { Об'ект - основа построения процесса / задачи }
  PTask = ^TTask;
  TTask = object(TObject)
    NTacts,    { текущее значение счетчика тиков / ТОЛЬКО ЧТЕНИЕ }
    Ticks : integer; {приоритет процесса (по умолчанию = 5)/ТОЛЬКО
ЧТЕНИЕ }
    Freeze   : boolean;  {True, если задача заморожена / ТОЛЬКО ЧТЕНИЕ }
    Stack,               { указатель на стек задачи / ТОЛЬКО ЧТЕНИЕ }
    TaskSSSP : pointer;  { текущее состояние задачи / ТОЛЬКО ЧТЕНИЕ }
    constructor Init;              { инициализация задачи }
    procedure Executer; virtual; { исполняемый метод задачи / }
                                   { перекрывается: всегда }
    procedure SetPriority(Prior : integer); { установка нового
приоритета задачи }
    procedure LockTV; { монопольное исполнение задачи / блокировка TV }
    procedure UnLockTV;            { разблокирование TV }
    destructor Done; virtual;      { завершение задачи }
  end;

  { Объект - основа построения многозадачной среды }
  PMultyTaskApp = ^TMultyTaskApp;
  TMultyTaskApp = object(TApplication)
    Tasks   : PCollection;         { коллекция процессов }
    constructor Init;              { инициализация среды }
    procedure Idle; virtual;       { переключатель процессов }
    destructor Done; virtual;       { выход из среды }
    procedure NewTask(ATask : PTask);   { порождение нового процесса }
    procedure PreTaskSw; virtual;       {Вызывается при порождении }
                                        { первого процесса }
    procedure PostTaskSw; virtual;      { Вызывается при окончании }
                                         { последнего процесса }
    procedure FreezeTask(ATask : PTask); { заморозка задачи }
    procedure UnFreezeTask(ATask : PTask); { разморозка задачи }
    procedure FreezeAll;           { заморозка всех задач системы }
    procedure UnFreezeAll;         { разморозка всех задач системы }
    procedure TermTask(ATask : PTask);  { уничтожение задачи }
    procedure TermAll;                  { уничтожение всех задач }
    procedure TaskError; virtual;       { перекрывается пользователем }
                                        { для реакции на ошибку }
  end;

const
  TaskStackSize : word    = 32*1024;   { размер стека фоновой задачи }
  TVAvailable   : boolean = true;      { признак монопольного       }
                                       { выполнения задачи          }
  MultyTaskApp  : PMultyTaskApp = nil; { указатель на PMultyTaskApp }

                          Implementation
Uses
   Drivers, Dos;                 { используемые модули }
var
  Old1C       : pointer;         { сохранение старого вектора 01Ch }
  SaveExitProc : pointer;        { сохранение старой точки выхода из}
                                 { программы }

type
  StackRecord =   { тип - стек задачи }
    record
      Size : Word;               { размер стека }
      Address : Pointer;     { реальный адрес указателя для захвата памяти }
    end;

{--------------------------------------}
              { Внешние подпрограммы }
{$L SWITCH.OBJ}
function  InitTaskStack(AStack, ATaskAddr, This : pointer) : pointer;
far; external;
   { формирование стека задачи }
function GoToTask(var TaskSP : pointer) : boolean; far; external;
   { переключение задачи }
procedure ClockInt(BP : Word); far; external;
   { обработчик 01Ch прерывания по таймеру }
{--------------------------------------}

              { описание полей TTask }

constructor TTask.Init;
   { захват и инициализация стека под задачу }
var
  StackRecPtr : ^StackRecord absolute Stack;
  POfs : Word absolute Stack;
  PTemp : Pointer;
  ActualSize : LongInt;
  TaskAddr : pointer;

begin
  Inherited Init;
  { размер с учетом записи для стека }
  ActualSize := LongInt(TaskStackSize)+6;
  Stack := nil;
  Ticks := 5;    { начальное значение приоритета }
  { А хватит ли памяти ? }

  if (MaxAvail >= ActualSize) and (ActualSize <= 65521) then begin
     GetMem(Stack, Word(ActualSize));
     PTemp := Stack;             {сохранение адреса}
     Inc(POfs, TaskStackSize); {сдвиг указателя на вершину стека }
     asm                    { доступ к виртуальному адресу @Executer }
       les bx,Self
       mov bx,es:[bx]
       mov ax,word ptr [bx + 0Ch]        { смещение }
       mov bx,word ptr [bx + 0Ch + 2]    { сегмент }
       mov word ptr TaskAddr,ax
       mov word ptr TaskAddr+2,bx
     end;
     TaskSSSP := InitTaskStack(Stack, TaskAddr, @Self);
     with StackRecPtr^ do begin  { запомним размер и начало стека }
       Address := PTemp;
       Size := Word(ActualSize);
     end;
  end;
end;

procedure TTask.Executer;
begin
  { Здесь располагается задача пользователя.
     Этот метод перекрывается всегда }
end;

procedure TTask.SetPriority(Prior : integer);
```

```
  {Устанавливает новое значение приоритета задачи.
     Большему значению Prior соответствует
     меньший приоритет задачи
     (начальное значение - 5) }
begin
  Ticks := Prior;
end;

procedure TTask.LockTV;
  { Для выполнения нереентерабельных процедур Turbo Vision
    и Turbo Pascal (Borland Pascal), типа New, Dispose, Intr ...
    в фоновой задаче.
    Блокирует выход из задачи.
    Используется только фоновыми процессами. }
begin
  TVAvailable := False;
end;

procedure TTask.UnLockTV;
  { Разблокирует основной процесс после исполнения
    нереентерабельных процедур.
    Используется после вызова LockTV }
begin
  TVAvailable := true;
end;

destructor TTask.Done;
  { удаление стека задачи при удалении задачи }
var
  StackRecPtr : ^StackRecord absolute Stack;
begin
  if Stack <> nil then
    with StackRecPtr^ do
      FreeMem(Address, Size);  { освобождение памяти из-под стека }
  inherited Done;
end;

{---------------------------------------}

              { описание полей TMultyTaskApp }

constructor TMultyTaskApp.Init;
 { инициализация среды }
begin
  inherited Init;    { инициализация TApplication }
  MultyTaskApp := @Self;  { установка указателя на TMultyTaskApp }
  Tasks := New(PCollection, Init(5,5));
             { инициализация коллекции задач }
end;

procedure TMultyTaskApp.Idle;
  { ************ переключатель задач *****************
    Переключение происходит в свободное от исполнения
    основного процесса время.
    Гарантируется монопольное выполнение
    нереентерабельных процедур основным процессом
    без дополнительных действий со стороны пользователя }
var
   T : PTask;  { указатель на текущую переключаемую задачу }
const
   CurTask  : Integer = 0;  { номер текущей задачи в коллекции задач }
   OldCount : Integer = 0;  { номер предыдущей задачи в коллекции задач
}
   function NextTask : pointer;
   var
     T : PTask;
     NT: integer;
   begin
     if CurTask >= OldCount then CurTask := 0;  { если текущая задача
                  выходит за границу коллекции задач,
                  то устанавливаем на первую задачу }
     NT := CurTask;   { запомним номер текущей задачи }
     repeat
       T := Tasks^.At(CurTask); { адрес текущей задачи }
       with T^ do
       begin
         if not Freeze then    { если задача не заморожена }
         begin
           inc(NTacts);        { следующий тик времени задачи }
           if NTacts = Ticks then
           begin                  { если достигли значения приоритета }
             inc(CurTask);    { переместимся на следующую задачу }
             NTacts := 0;     { для вычисления нового тика вызова }
             NextTask := T;   { вернем указатель на задачу }
             exit;
           end;
         end;
       end;
       inc(CurTask);          { продолжаем сканирование по коллекции задач }
       if CurTask >= OldCount then CurTask := 0;
     until CurTask = NT;      { сканируем до совпадения с исходным номером
задачи}
     NextTask := nil;         { если ни у одной задачи не подошло время }
                              { для переключения, то вернем NIL }
   end;

begin
  inherited Idle; { метод Idle объекта TApplication }

  if (OldCount =  0) and (Tasks^.Count <> 0) then PreTaskSw else
  if (OldCount <> 0) and (Tasks^.Count =  0) then PostTaskSw;
     { проверка условия вызова PreTaskSw и PostTaskSw }
  OldCount := Tasks^.Count;          { сохранение количества задач }

  if OldCount <> 0 then                  { если коллекция задач не пуста }
  begin
    T := NextTask;                   { запрос на очередную задачу }
    if T <> nil then                  { если она нашлась, то }
      if GoToTask(T^.TaskSSSP)  { переключение на задачу }
        then Tasks^.Free(T);      { если она завершилась - }
                                  { закроем задачу и удалим из коллекции }
  end;
end;

procedure TMultyTaskApp.NewTask( ATask : PTask );
  { включение новой задачи в коллекцию }
begin
   if (ATask = nil) or   { если задача = nil }
     (ATask^.Stack = nil) then TaskError   { или она без стека, то
                                             ошибка }
   else Tasks^.Insert(ATask); { иначе включаем в коллекцию }
end;

procedure TMultyTaskApp.PreTaskSw;
  { Вызывается при порождении первого процесса.
    Перекрывается пользователем для установки параметров
    свой системы, например разрешение/запрещение
    некоторых команд EnableCommands([])/DisableCommands([])
    или любых иных действий }
begin
end;

procedure TMultyTaskApp.PostTaskSw;
  { Вызывается при окончании последнего процесса.
    Перекрывается пользователем, аналогично PreTaskSw }
begin
end;


procedure TMultyTaskApp.FreezeTask(ATask : PTask);
  { Вызывается для временного замораживания процесса (задачи)/
    запрещения переключения на данную задачу }
begin
  if ATask <> nil then ATask^.Freeze := true;
end;

procedure TMultyTaskApp.UnFreezeTask(ATask : PTask);
  { размораживание процесса (задачи) }
begin
```

```
    if ATask <> nil then ATask^.Freeze := false;
  end;

procedure TMultyTaskApp.TermTask(ATask : PTask);
  { удаление (терминирование) задачи и
    исключение из коллекции задач }
begin
  Tasks^.Free(ATask);
end;

procedure TMultyTaskApp.FreezeAll;
  { замораживание всех задач }
  procedure FreezeItem(Item : PTask); far;
  begin
    Item^.Freeze := true;
  end;
begin
  Tasks^.ForEach(@FreezeItem);
end;

procedure TMultyTaskApp.UnFreezeAll;
  { размораживание всех задач }
  procedure UnFreezeItem(Item : PTask); far;
  begin
    Item^.Freeze := false;
  end;
begin
  Tasks^.ForEach(@UnFreezeItem);
end;

procedure TMultyTaskApp.TermAll;
  { удаление (терминирование) всех задач }
begin
  Tasks^.FreeAll;
end;

procedure TMultyTaskApp.TaskError;
  { Виртуальный метод вызывается при ошибке включения
    новой задачи в систему.
    Перекрывается пользователем для реакции на ошибку }
begin
end;

destructor TMultyTaskApp.Done;
  { завершение многозадачной среды }
begin
  Dispose(Tasks, Done);  { удаляем коллекцию задач }
  inherited Done;        { вызываем TApplication.Done }
end;

{-------------------------------------}

procedure MTExit;
begin
  ExitProc := SaveExitProc;  { перед выходом вернем адрес выхода }
  SetIntVec($1C, Old1C);     { и старое значение вектора прерывания }
end;

begin
  SaveExitProc := ExitProc;  { сохраним адрес выхода из программы }
  ExitProc := @MTExit;       { и установим свой }
  GetIntVec($1C, Old1C);     { сохраним вектор прерывания по таймеру }
  SetIntVec($1C, @ClockInt); { и установим свой }
end.
```

■

Листинг 2

```
;    Вспомогательный модуль для
; Pascal - модуля MultTask.pas (.tpu)
;       для создания
;    псевдомногозадачной среды
;      (фоновых процессов)
;        в Turbo Vision
;
; транслятор:
;    Borland TASM v3.2
;
;  разработанно:
;    Миллер Станислав Юрьевич
;    Москва / 1994
;
      TITLE  Switch
DATA  SEGMENT  BYTE PUBLIC
  EXTRN TVAvailable : byte     ; признак доступности выхода из задачи
DATA  ENDS
CODE  SEGMENT BYTE PUBLIC
      ASSUME  CS:CODE,DS:DATA
      PUBLIC  InitTaskStack, GoToTask, ClockInt
SProom:  dw 0          ; для сохранения значения SP основного процесса
SSroom:  dw 0          ; для сохранения значения SS основного процесса
RetTaskEn: db 0        ; признак возврата из фоновой задачи
; function InitTaskStack(Stack, TaskProc, This : pointer) : pointer;
;    { формирование стека задачи }
;
InitTaskStack PROC FAR
  push  bp             ; сохраним BP для Pascal
  mov   bp,sp
  les   di,[bp+14]     ; es:di указывает на начало (верхушку) Stack
  sub   di,2           ; на начало стека (декремент и запись)
  std                  ; флаг направления - на уменьшение (dec)
                       ; начинаем заполнять стек задачи
  mov   ax,[bp+8]      ; seg @Self
  stosw                ; аналог команды - push seg Self
  mov   ax,[bp+6]      ; offset @Self
  stosw                ; аналог команды - push offset Self
                       ; поместим в стек адрес возврата по окончанию
  mov   ax,cs          ; задачи (процесса) - @EndTask,
  stosw                ; которая находится в этом же сегменте
  mov   ax,offset EndTask
  stosw
  pushf                ; используется текущее значение флагов
  pop   ax             ; для задачи (фонового процесса)
  stosw                ; Flags for iret
  mov   ax,[bp+12]     ; адрес задачи, на который впоследствии
  stosw                ; произойдет переход командой iret
  mov   ax,[bp+10]     ; сначала - сегмент, теперь - смещение
  stosw
  xor   ax,ax      ; подготавливаем остальные регистры для задачи
  mov   cx,7       ; начальные значения - нулевые
  rep stosw        ; AX, BX, CX, DX, SI, DI, BP  (7 шт.)
  mov   ax,ds      ; подготовим значения для ES и DS задачи
  stosw            ; ES = seg DATA
  stosw            ; DS = seg DATA
  pop   bp         ; восстановим BP для Pascal
  mov   dx,es      ; вернем сегмент начала стека задачи
  mov   ax,di      ; ...и смещение
  add   ax,2       ; откорректируем указатель, т.к. стек указывает
                   ; на текущий адрес
  retf  12
InitTaskStack ENDP
; function GoToTask(var CurStack : pointer) : boolean
;    { true - задача  завершена }
;    { переключение задачи }
GoToTask PROC  FAR
  push  bp             ; сохраняем BP основного процесса
  mov   bp,sp
  les   di,[bp+6]      ; загрузим указатель на @CurStack
  push  di                 ; сохраним указатель на @CurStack в стеке
  push  es
  push  ds             ; сохраняем DS основного процесса
  mov   ax,es:[di]     ; сохраним значение SP задачи (фонового процесса)
  mov   bx,es:[di+2]   ; сохраним значение SS задачи (фонового процесса)
  cli                  ; для переустановки SS:SP запретим прерывания
  mov   word ptr cs:SSroom,SS   ; сохраним SS:SP основного процесса
  mov   word ptr cs:SProom,SP
  mov   ss,bx                       ; новое значение SS:SP
  mov   sp,ax
  sti
```

```
   mov   byte ptr cs:RetTaskEn,1     ; установим признак возврата
                                     ; задачи из прерывания
   pop   ds
   pop   es
   pop   bp
   pop   di          ; восстановим регистры задачи из стека
   pop   si
   pop   dx
   pop   cx
   pop   bx
   pop   ax
   iret              ; перейти на задачу (фоновый процесс)
GoToTask ENDP
; procedure ClockInt(BP : Word);
;   { обработчик 01Ch прерывания по таймеру }
;
ClockInt       PROC    FAR   ; interrupt
   push  ax
   push  bx
   push  cx
   push  dx
   push  si          ; сохраним регистры CPU
   push  di
   push  bp
   push  es
   push  ds
;
   mov   ax,seg DATA
   mov   ds,ax                       ; DS установить на сегмент DATA
   cmp   byte ptr TVAvailable,0      ; если выход из задачи запрещен,
   je    ExitInt                     ; то на выход из прерывания
   cmp   byte ptr cs:RetTaskEn,0     ; проверим, а находимся ли мы
   je    ExitInt                     ; в фоновой задаче
;                                    ; если нет - то на выход
ReturnTask:
   xor   cx,cx       ; false - если задача еще не завершена
EndGoToTask:
   mov   byte ptr cs:RetTaskEn,0   ; возвращаемся из фоновой задачи
   mov   al,20h
   out   20h,al          ; команда - конец прерывания
   cli                   ; при изменении SS:SP - запретим прерывания
   mov   bx,SS           ; сохраним SS задачи
   mov   ax,SP           ; сохраним SP задачи
   mov   SS,word ptr cs:SSroom     ; установим SS:SP основного
   mov   SP,word ptr cs:SProom     ; процесса
   sti                   ; после установки SS:SP - разрешим прерывания
   pop   ds              ; восстановим Pascal's DS
   pop   es              ; восстановим ES:DI - указатель на указатель
   pop   di              ; текущего значения SS:SP задачи
   mov   es:[di],ax      ; новое значение SS:SP задачи
   mov   es:[di+2],bx
   mov   ax,cx           ; признак завершенности задачи в CX
   pop   bp              ; восстановим Pascal's BP
   retf  4               ; вернемся к выполнению основного процесса
ExitInt:
   pop   ds
   pop   es
   pop   bp
   pop   di
   pop   si              ; в случае невыполнения условия выхода из
   pop   dx              ; задачи восстановим регистры
   pop   cx               ; и...
   pop   bx
   pop   ax
   iret                  ; ...выйдем из прерывания
ClockInt       ENDP
EndTask:
   mov   cx,1            ; true - если задача завершена
   jmp short EndGoToTask         ; выходим из задачи
CODE   ENDS
END
```

Листинг 3

```
{       Программа для демонстрации
        псевдомногозадачной среды
          (фоновых процессов)

              в Turbo Vision

       для транслятора Turbo Pascal
            и Borland Pascal
 для реального режима работы процессора 80x86

 протестированно:
   CPU : 80286, 80386DX, 80486DX
   RAM : 1Mb, 4 Mb
   OS  : DOS 5.0
   TV  : Turbo Vision v2.0
   PAS : Borland Pascal v7.0

 разработанно:
   Миллер Станислав Юрьевич
   Москва / 1994
}
uses
 Crt,
 Objects,
 App,
 Drivers,   { включаемые модули Turbo Vision }
 Views,
 Menus,
 MultTask;  { модуль для создания псевдомногозадачной
              среды (фоновых процессов) в Turbo Vision }
const
 cmNewTask     = 100;   { список используемых комманд }
 cmTermTask    = 101;
 cmFreezeTask  = 102;
 cmUnFreezeTask = 103;
 Sym : array[0..3] of char = '▒▓ ';  { символ - заполнитель
                                       для DEMO - окон }
type
 TFonApp = object(TMultyTaskApp)
  constructor Init;
  procedure InitMenuBar; virtual;
  procedure HandleEvent(var Event: TEvent); virtual;
  procedure PreTaskSw; virtual;   { при порождении первого процесса }
  procedure PostTaskSw; virtual;  { при окончании последнего }
                                  { процесса }
  procedure OpenTask;              { открыть задачу (процесс) }
  procedure TermTask;             { закрыть задачу (процесс) }
  procedure FreezeTask;            { заморозить все задачи }
  procedure UnFreezeTask;         { разморозить все задачи }
 end;

 PDemoWindow = ^TDemoWindow;       { DEMO-окна для оживления процесса }
 TDemoWindow = object(TWindow)     { псевдомногозадачности }
  FillP : byte;                     { символ - заполнитель окна }
  Master : PTask;              { указатель на второго владельца (PTask) }
  constructor Init(AMaster : PTask);   { (первый - DeskTop) }
  procedure HandleEvent(var Event : TEvent); virtual; { для }
                                  { отслеживания   команды cmClose }
  procedure Draw; virtual;        { вычерчивание окна }
 end;

 PDemoTask = ^TDemoTask;           { объект для демонстрации TTask }
 TDemoTask = object(TTask)
  PW : PDemoWindow;                { окно фоновой задачи }
  constructor Init;
  procedure Executer; virtual;     { собственно задача }
  destructor Done; virtual;
 end;

{---------------------------------}

constructor TFonApp.Init;
begin
 inherited Init;
```

```
    TaskStackSize := 1024;   { размер стека фоновой задачи }
    DisableCommands([cmTermTask, cmFreezeTask, cmUnFreezeTask]);
        { запретим эти команды при отсутсвии задач (начальные
условия)}
end;

procedure TFonApp.InitMenuBar;
var R:TRect;
begin
 GetExtent(R);     { стандартная инициализация меню }
 R.B.Y := R.A.Y+1;
 MenuBar := New(PMenuBar, Init(R, NewMenu(
   NewSubMenu('~F~ile', hcNoContext, NewMenu(
     NewItem('~N~ew Task', 'F3', kbF3, cmNewTask, hcNoContext,
     NewItem('Free~z~e All Task','', kbNoKey, cmFreezeTask,
hcNoContext,
     NewItem('~U~nFreeze All Task','', kbNoKey, cmUnFreezeTask,
hcNoContext,
     NewItem('~T~erminate All Task', 'Alt-F4', kbAltF4, cmTermTask,
hcNoContext,
      NewLine(
     NewItem('E~x~it', 'Alt-X', kbAltX, cmQuit, hcNoContext,
     nil)))))),
   NewSubMenu('~W~indow', hcNoContext, NewMenu(
     StdWindowMenuItems(
     nil)),
   nil)))));
end;

procedure TFonApp.HandleEvent(var Event: TEvent);
begin
  if Event.What = evCommand then
  begin
    case Event.Command of
      cmNewTask : OpenTask;              { отработка наших комманд }
      cmTermTask : TermTask;             { касающихся псевдо- }
      cmFreezeTask : FreezeTask;         { многозадачного режима }
      cmUnFreezeTask : UnFreezeTask;     { работы }
    end;
  end;
  inherited HandleEvent(Event);

end;

procedure TFonApp.PreTaskSw;
begin
  EnableCommands([cmTermTask, cmFreezeTask, cmUnFreezeTask]);
    { в случае порождения первой команды разрешим эти команды }
end;

procedure TFonApp.PostTaskSw;
begin
  DisableCommands([cmTermTask, cmFreezeTask, cmUnFreezeTask]);
    { в случае удаления последней команды запретим эти команды }
end;

procedure TFonApp.OpenTask;
begin
 NewTask(New(PDemoTask, Init));  { новая задача (фоновый процесс) }
end;

procedure TFonApp.TermTask;
begin
  TermAll;      { удалим все задачи }
end;
procedure TFonApp.FreezeTask;

begin
  FreezeAll;    { заморозим все задачи }
end;

procedure TFonApp.UnFreezeTask;
begin
  UnFreezeAll;  { разморозим все задачи }
end;

{-----------------------------------}
```

```
constructor TDemoTask.Init;
begin
  inherited Init;
  if Stack = nil then exit;  { если стек не разместился в heap - выход }
  PW := New(PDemoWindow, Init(@Self));  { создадим окно }
  MultyTaskApp^.InsertWindow(PW);       { вставим его в DeskTop }
end;

procedure TDemoTask.Executer;
var I : integer;

begin
  if PW = nil then exit;          { если окна нет - выход }
  repeat
    Delay(100);                   { бурная деятельность }
    inc(PW^.FillP);               { следующий заполнитель }
    PW^.FillP := PW^.FillP and 3;
    LockTV;                       { перерисовка окна в монопольном режиме }
    PW^.Draw;                     { нарисуем окно }
    UnLockTV;                     { разблокируем Turbo Vision }
  until false;                     { работаем "без конца" }
end;

destructor TDemoTask.Done;
begin
  PW^.Close;                      { в случае терминирования задачи }
  inherited Done;                 { закроем окно }
end;

{-----------------------------------}

constructor TDemoWindow.Init(AMaster : PTask);
var
   R : TRect;
begin
  R.Assign(Random(8)+2, Random(5),      { стандартное создание окна }
         Random(60)+20, Random(12)+10);
  inherited Init(R,'Demo Window',0);
  Master := AMaster;                  { указатель на второго владельца }
  Options := Options or ofTileable;
end;

procedure TDemoWindow.Draw;
var
  Color: Byte;
  Y: Integer;

begin
  inherited Draw;                     { вычерчивание рамки и прочее }
  Color := 1;
  for Y := 1 to Size.Y - 2 do         { заполним все окно заполнителем }
    WriteChar(1, Y, Sym[FillP], Color, Size.X-2);
end;

procedure TDemoWindow.HandleEvent(var Event : TEvent);
begin
  if (Event.What = evCommand) and (Event.Command = cmClose) then
  begin    { перехватим команду cmClose }
    MultyTaskApp^.TermTask(Master);   { сообщим второму владельцу
(TTask) }
    ClearEvent(Event);      {о попытке пользователя терминировать окно }
                            { реакция - удаление задачи из колекции задач
                            (окно закроется в методе TTask.Done) }
  end else
      inherited HandleEvent(Event);
end;

var
   MyApp : TFonApp;   { далее по стандартному Turbo Vision }

begin
  MyApp.Init;
  MyApp.Run;
  MyApp.Done;
end.                                                          ■
```

CASE - технология:
SWS-программирование без программистов!

г. Москва, ВГРТК «Останкино», тел. 215-10-24
Исаева Татьяна Юрьевна, старший программист службы автоматизации.
- Мы пользуемся системой больше двух лет. Считаем, что это очень мощная система. В настоящий момент разрабатывается прикладной комплекс обработки документации для телецентра. SWS сокращает время на разработку и трудозатраты по самой разработке. Она дисциплинирует пользователя во время работы, т.к. предъявляет жесткие требования при создании БД и связей между файлами, заставляет пользователя строить оптимальную систему файлов БД, что приводит к экономии памяти и увеличению быстродействия. Конечно, как и у любого другого продукта, у SWS есть определенный ряд недостатков, но есть и самое главное достоинство - чисто западный подход «Монолога» в вопросах поддержки пользователей и разработки обновлений

г. Москва, АО «Аргументы и факты», Служба распространения, тел. 923-68-59, 923-34-65
Желтов Олег Владимирович, директор службы распространения.
- Мы пользуемся системой 2 года и пользуемся, в основном, косвенно. Наши партнеры по Украине - фирма «Монолог» - внедрили у нас большое количество прикладных задач, разработанных с помощью их инструментального средства, и эти прикладные системы, в некотором смысле, позволили даже расширить сферу воздействия нашей службы. На SWS реализовано большое количество нестандартных постановок, связанных со спецификой именно газетно-книжного сбыта, делопроизводства и бухгалтерской отчетности, что говорит об универсальности подхода и инструментария. Все прикладные программы работают хорошо и очень быстро модифицируются, если это необходимо.

г. Москва, АО «Артифлекс», тел. 252-48-81
Видерман Михаил Феликсович, президент.
- С помощью системы на нашей фирме ведется разработка двух конкретных прикладных проектов в Москве. И если бы не SWS, нам пришлось бы искать нечто подобное среди западных продуктов, ибо разрабатывать и внедрять такие проекты на языках третьего поколения мы бы не стали, а значит были бы вынуждены отставать по времени. Системой довольны и особенно довольны поддержкой «Монолога». Уже с нашей помощью система внедрена в Москве в АО «Новые технологии», АОЗТ «Интерьер» и на фирме «К-8».

г. Москва, Федеральное дорожно-строительное управление при Министерстве обороны Российской Федерации, тел. 296-37-56, 915-41-20
Аршенов Виталий Кондратьевич.
- С системой работаю уже год. За последние 4 месяца с помощью SWS самостоятельно «оживил» локальную сеть всего нашего управления. Знаком со многими продуктами, но предпочтение отдаю именно этой системе, т.к. она удовлетворяет всем моим требованиям к инструментарию. Считаю, что у SWS большое будущее и она может завоевать рынок СНГ.

г. Москва, ассоциация промбанков «Россия», тел. 281-87-97
Шмелев Геннадий Валентинович, старший программист.
- С системой работаю 2 года. На ней написано и внедрено более 10 продуктов, в том числе «Справочник по банкам России». У SWS есть будущее, она на рынке СНГ может выступать как готовый продукт. По отношению к другим системам, таким как Oracle, FoxPro, она гораздо дешевле при почти аналогичных свойствах, оригинальна и очень проста в освоении. Надо отдать должное «Монологу» за поддержку своего продукта и работу с пользователями. Для СНГ это почти чудо.

г. Москва, Медунов Александр Львович - частный пользователь, тел. 287-69-51
- Считаю, что система в настоящий момент очень актуальна на рынке СНГ. У нее есть будущее, т.к. она позволяет быстро создавать программный продукт, обеспечивает быструю модификацию существующего прикладного программного продукта и внесение изменений и дополнений в структуру БД. Система позволяет также проектировать БД.

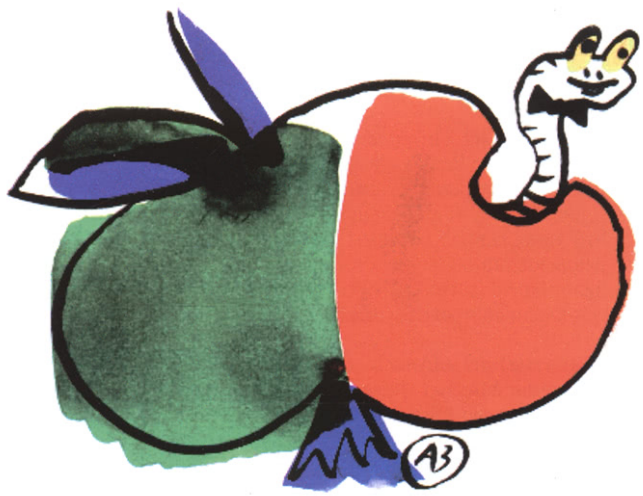
Украина, г. Киев, НИИ Автоматизированных систем в строительстве, Комитет по делам градостроительства и архитектуры, тел. 271-35-40, 244-05-63
Возняк Владимир Владимирович, зав. отделом.
- Система проходила у нас частичное тестирование и работает 3 года. С ее помощью решено много прикладных задач, и она внедрена на 5 предприятиях Украины. Во-первых, это действительно завершенное как продукт инструментальное CASE-средство, что само по себе для СНГ - сенсация, а во-вторых, вполне конкурентоспособный товар. Если с SWS работать серьезно - проектировать серьезные прикладные комплексы - она очень быстро окупается.

Украина, г. Хмельницкий, войсковая часть А 3779, тел. 14-73-72
Шушунов Владимир Васильевич, старший программист.
- Мы пользуемся системой с октября 1994 года. Считаю, что ее необходимо повсеместно внедрять, т.к. она позволяет продуктивно работать даже неквалифицированному пользователю-проектировщику. И вообще, пользователям SWS необходимо чаще встречаться, устраивать семинары. Возможна организация единомышленников в определенный клуб любителей внедрять ПО, а не просто разрабатывать его до пенсии, с обменом мнений, опытом. SWS для ее пользователей уже стала неким символом промышленного программирования. И это сегодня очень актуально.

Украина, Одесская обл., г. Измаил, Украинское Дунайское пароходство, тел. 2-07-36
Дуков Георгий Степанович, программист.
- Пользуюсь системой год. Наша организация не полностью перешла на SWS, т.к. нет сервера, поэтому нет возможности полного использования в сетевом режиме. Считаю, что у этого продукта - большое будущее. Очень рад, что «Монолог» - украинская фирма.

Жизнь на Макинтоше: по страницам новой книги

М. Трофимов



Краткий авторский пересказ главы, посвященной программированию на Макинтоше, из новой книги московской фирмы «МакЦентр». В качестве примера использована широко известная модель клеточного автомата — игра «ЖИЗНЬ».

Часть 1

Этой статьей редакция открывает новую рубрику «Программирование на Макинтоше», которая, надеемся, будет особенно полезной читателям «Монитора». Дело в том, что сейчас в нашей стране наблюдается резкий подъем спроса на эту популярную во всем мире компьютерную платформу. Так, по сведениям фирмы «МакЦентр» — одного из лидеров в данном секторе рынка, — объем продаж удваивается каждые четыре месяца. Однако издательская деятельность сильно отстает — нет не только оригинальной, но даже переводной литературы о Маке. Заполнить этот пробел должна книга «Компьютер Макинтош — пособие для пользователя», подготовленная в фирме «МакЦентр» коллективом авторов. Это первая русская книга о Макинтоше. Помимо вопросов программирования, в книге будет приведена самая свежая информация о компьютерах фирмы «Apple Computer, Inc.» и программном обеспечении к ним, операционной системе, издательских системах, а также краткий вводный курс, позволяющий в считанные часы освоить работу оператора на компьютерах Макинтош. К книге будут приложены дискеты с наиболее интересными программами, полезными в практической работе.

Издатель книги — фирма «МакЦентр», один из первых у нас дилеров фирмы «Apple Computer, Inc.» — за время своей деятельности накопил большой опыт работы, в том числе и маркетингового характера с учетом особенностей нашей страны и специфических требований пользователей, который также найдет отражение в новой книге.

Заказы на книгу можно направлять в фирму «МакЦентр» по адресу: 117191 Москва, Гамсоновский пер., 9-34, тел. (095) 955-2720 (с 9 до 18 часов). Предварительной оплаты не требуется.

Предлагаемая статья — краткий авторский пересказ главы, посвященной программированию на Макинтоше, из будущей книги. Автор использовал свой опыт работы в фирме «Visible Software» (Делавер, США) по созданию версии Pascal системы Dr Pascal for Macintosh, предназначенной для обучения студентов программированию на языке Pascal. Версия системы Dr Pascal, о которой мы уже рассказывали читателям на страницах журнала [1], для других платформ (MS-DOS, OS, VAX VMS) несколько лет активно используется во многих учебных заведениях и завоевала популярность не только в США, но и во всем мире — от Австралии до России, где уже появились ее первые версии.

Программирование на Маке

На Макинтоше реализован удобный графический интерфейс, поэтому пользователю-непрограммисту работать на нем значительно проще, чем, скажем, на IBM PC в MS-DOS — не надо запоминать написание команд (тем более, что они на Маке локализуются — в случае России русифицируются), однако обеспечить поддержку в прикладной программе этого интерфейса программисту, особенно на первых шагах, может оказаться нелегко. Даже у опытного в работе с другими компьютерами, скажем, с IBM PC, специалиста при первом знакомстве с Маком может сложиться впечатление, что та же задача решается средствами MS-DOS проще, во всяком случае, в смысле организации ввода-вывода. Такое мнение было бы поверхностным. Попробуйте даже для простенькой задачи (типа той, которой мы займемся здесь позже) запрограммировать в DOS аналогичный, не сильно уступающий по возможностям интерфейс с пользователем, и вы убедитесь, что на Маке это сделать проще. Единственное, что может быть действительно труднее, это сделать слишком бедный интерфейс, как это часто бывает

на IBM PC в наспех «сляпанных» программах, то есть сделать плохую, недружелюбную программу на IBM PC, может быть, действительно проще.

И еще, стандартные подходы и стандартные решения приводят к тому, что разные программы на Маке более похожи по внешнему виду, чем на PC. И в этом еще одно принципиальное достоинство — похожие по интерфейсу программы пользователю легче освоить. Даже для нескольких знакомых программ было бы удобнее, чтобы одни и те же клавиши и их комбинации были ответственны за одни и те же действия, например, чтобы из любой программы можно было выйти по команде Alt-Q. К сожалению, на IBM PC мы видим обратную картину: в некоторых программах выход Alt-Q, в других Alt-X, в третьих F10...; в одних линейка основного меню сверху, в других снизу и так далее. Но за удобства будущих пользователей программисту приходиться платить. Тут прежде всего надо понять общие принципы программирования на Макинтоше, а уж детали можно посмотреть в документации.

Мы не ставим своей целью сравнение компьютерных платформ. И если где-либо здесь и упоминается IBM PC или MS-DOS, то только потому, что сегодня у многих российских программистов есть опыт работы на IBM PC и такие сравнения могут облегчить им понимание. Точно так же еще несколько лет назад в советских книгах, посвященных IBM PC, можно было найти противопоставления коллективным машинам «Единой Системы» (ЕС). Да, въелась мечта о некой Единой Унифицированной Серии Компьютеров.

Действительность же такова, что нет на нынешний момент единой, Истинно Верной, принятой всеми инстанциями компьютерной платформы, а есть ряд успешно соревнующихся друг с другом производителей. Важно, что их не так много, значит, о хаосе говорить не приходится, важно, что не один, значит, есть конкуренция, гарантирующая дальнейший прогресс. Важно, что производители делают шаги навстречу друг другу, например, часть этого текста набиралась на IBM PC, а часть на Маке, в зависимости от того, какой компьютер в это время был свободен от других задач. Впоследствии файлы переносились и объединялись в течение считанных секунд. А рабочие распечатки текстов делались то с Мака, то с PC на принтере Epson FX-1000: когда с PC — принтер подключался напрямую к порту, а когда с Мака — через маленькую «коробочку», укрепленную на конце кабеля, название «коробочки» — «Grappler 9 Pin», и делают ее на фирме «Orange Micro» (США).

Все это мы рассказываем здесь для того, чтобы у читателя не сложилось неверного представления о непреодолимой пропасти между различными компьютерными платформами и чтобы читатель, привыкший к иным платформам и иным подходам, был бы готов посмотреть на вещи шире и, может быть, пересмотреть в чем-то свои взгляды. Даже если вы и не сделали еще свой выбор, вам следует для расширения кругозора обратить свои взгляды на Мак и попробовать поработать на нем. Всего лишь первое практическое знакомство способно поколебать многие «незыблемые» убеждения.

При этом мы не помышляем навязать пользователю ту или иную концепцию, можно знать обе и работать как в той, так и в другой. Сейчас в условиях российского рынка для многих программистов, по-видимому, будет характерен именно такой прагматичный подход. Думается, что это разумно, поскольку, судя по международной периодике, посвященной вопросам найма, в том числе и иностранных специалистов, спрос на программирование для Мака иногда превышает спрос для иных платформ, в частности, для PC.

Операционная система, языковые среды и документация

Здесь и далее речь пойдет об операционной системе System 7.0.1, наиболее распространенной сейчас.

Детали, нужные программисту компьютера Макинтош, описаны в многотомном документе «Inside Macintosh» («Apple Computer, Inc.», «Addison-Wesley Publishing Company»). Там же приводятся примеры решения типовых задач, возникающих при программировании интерфейса, в виде коротких листингов на языке Pascal. Обращение к системным ресурсам на Pascal выглядит как вызов стандартной процедуры или функции. Всего таких стандартных процедур несколько сотен. Поэтому для удобства пользования они сгруппированы по функциональному назначению (работа с файлами, окнами, меню, управление оперативной памятью). Эти группы — компоненты операционной системы — называются *Менеджерами (Manager)*. А в число языковых средств входят интерфейсные библиотеки для связи с компонентами операционной системы.

Мы использовали Pascal, входящий в состав инструментального пакета MPW (Macintosh Programmer's Workshop), поставляемого фирмой «Apple Computer, Inc.», однако у тех, кто захочет воспроизвести листинги из этой статьи в среде Think Pascal «SYMANTEC», не возникнет особых проблем, тем более, что в наборе инструментов Think Pascal есть специальная утилита для трансляции исходных текстов MPW Pascal в исходные тексты Think Pascal, особых трудностей не возникнет также и у тех, кто захочет перенести приводимый здесь пример на язык C.

В качестве примера возьмем широко известную модель клеточного автомата — игру **«Жизнь»**, изобретенную Джоном Хортоном Конвеем.

Правила жизни

Запрограммируем на Маке эту игру при следующих условиях. Представим себе некую «пустыню» в виде прямоугольной решетки с квадратными ячейками, свернутой в тор (**жизнь** на поверхности тора). Каждая ячейка может быть либо пустой, либо заполненной. Заполненную ячейку будем называть клеткой. Согласно [2]:

1. Соседями клетки считаются все клетки, находящиеся в восьми ячейках, расположенных рядом с данной по горизонтали, вертикали или диагонали.
2. Если у клетки меньше двух соседей, она погибает от одиночества. Если клетка имеет больше трех соседей, она погибает от тесноты.
3. Если рядом с пустой ячейкой окажется ровно три соседние клетки **жизни**, то в этой ячейке рождается новая клетка.
4. Гибнущая клетка может способствовать рождению новой, но рождающаяся клетка не может воскресить гибнущую, и гибель одной клетки не может предотвратить гибель другой.

Первоначальная конфигурация вводится вручную с помощью специализированного графического редактора (Редактора **жизни**) или считывается из указанного пользователем файла (Документа **жизни**). Развертка тора **жизни** с живущими на нем клетками отображается на экране и обновляется при смене поколений.

Предусматриваются возможности прервать ход **жизни**, записать полученную конфигурацию в файл (Документ **жизни**) и выйти из программы.

Приступая к решению этой задачи, в целях оптимизации для сокращения перебора можно было бы сразу пойти нетривиальным путем, например, как это предложено в [2]. Однако, поскольку мы собираемся говорить об интерфейсе, выберем путь попроще.

С чего начинается жизнь?

С исходного файла ресурсов.

Одно из отличий между Макинтошами и другими компьютерами состоит в способе, которым Маки управляются с ресурсами (пиктограммы, окна, диалоги и так далее). В мире Мака ресурсы отличаются от данных (примером данных может быть текст этой статьи в файле текстового редактора).

Для подавляющего большинства компьютерных платформ файл состоит из последовательности байтов. *На Макинтоше файл состоит из двух частей* и содержит две последовательности байтов — *часть данных (data fork)* и *ресурсную часть (resource fork)*. Файл может содержать либо только часть данных, либо только ресурсную часть, либо и то и другое (рис. 1).

Ресурсы классифицируются по типу. Каждый тип имеет свое собственное имя, которое состоит из четырех символов. Хотя допустимы любые символы, обычно используют прописные и заглавные буквы, цифры, знаки препинания и пробелы.

Все стандартные типы ресурсов имеют стандартные имена. Например, «WIND» для окон, «MENU» для меню и так далее. Код прикладной программы также записывается редактором связей в ресурсную часть исполняемого файла (application) в качестве ресурса специального типа «CODE». Помимо стандартных типов ресурсов пользователь может определять и собственные типы, присваивая им уникальные имена.

Для работы с ресурсами имеются специальные инструментальные программы. Это ResEdit («Apple Computer, Inc.»), а также Rez и DeRez в составе MPW. *ResEdit* — это инструмент для редактирования ресурсной части любого файла. Работа со стандартными типами ресурсов ведется с помощью специальных диалоговых шаблонов, причем пользователь видит создаваемый ресурс, например, пиктограмму для прикладной программы, в виде, близком к естественному (рис. 2).

Инструментальная программа *DeRez* предназначена для трансляции ресурсной части файлов в текстовое представление на специальном языке. Примером такого представления может служить файл Life.r (листинг 1). С помощью другой программы, *Rez*, этот текстовой файл может быть перетранслирован обратно в соответствующие коды (не путать слово «коды» с командами процессора), которые помещаются в ресурсную часть выходного файла программы Rez (рис. 3).

Обычно, приступая к работе по написанию программы для Мака, начинают с описания ресурсов либо с помощью ResEdit, либо с помощью текстового редактора. В первом случае будет получен файл ресурсов, который затем связывается редактором связей с кодом создаваемой программы, во втором случае получается исходный текст для последующей трансляции с помощью Rez. Программировать ресурсы в ResEdit обычно удобнее (в силу наглядности), кроме того, можно скопировать и при необходимости модифицировать какие-либо ресурсы из другой своей программы — непосредственно из исполняемого файла! В то же время такую работу трудно документировать — распечатывать каждый раз экран для каждого ресурса, как это сделано для рисунка 2, слишком хлопотно, не говоря уже о неоправданном расходе бумаги и картриджа. Поэтому в книгах обычно приводят исходные коды ресурсов в текстовом виде, как это сделано в листинге 1. Кроме того, в языке Rez (Rez language) имеется целый ряд полезных средств, недоступных при работе в ResEdit. С помощью DeRez эти два подхода можно комбинировать, скажем, начать работу в ResEdit, оттранслировать созданный файл и продолжить работу на языке Rez.

Итак, посмотрим, как описаны ресурсы **жизни**.

Ресурсы жизни

Прежде всего каждому ресурсу нужно присвоить идентификационный номер, или просто идентификатор. Идентификатор должен быть уникален внутри данного типа ресурсов. Например, будет ошибкой определить два окна с одним и тем же идентификатором 128, но можно

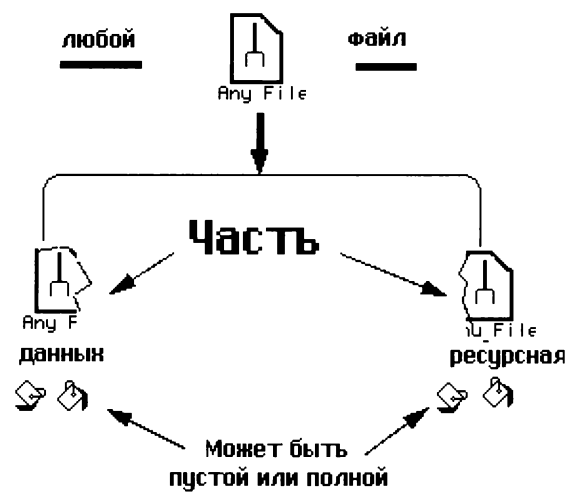


Рис. 1. Две части файла в файловой системе Макинтоша

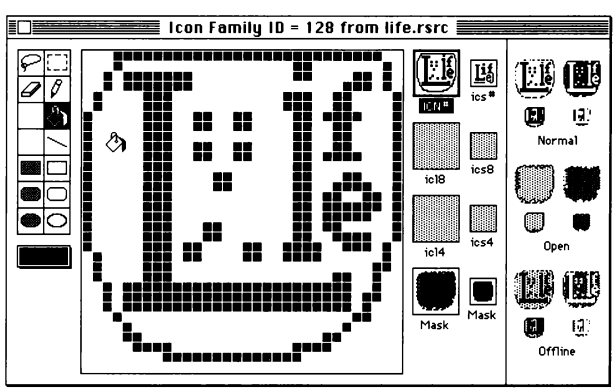


Рис. 2. ResEdit

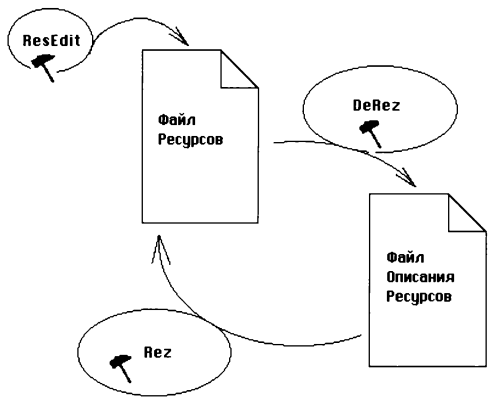


Рис. 3. Инструменты программирования ресурсов: ResEdit, Rez и DeRez

присвоить этот идентификатор окну и диалогу. Идентификатором может быть любое целое число типа integer, но «Apple Computer» рекомендует использовать для идентификаторов пользователя числа в интервале 127..32767 (при этом для некоторых стандартных типов ресурсов существуют дополнительные ограничения). Дело в том, что многие идентификаторы из других интервалов зарезервированы.

Для получения системой информации о программе используется также специальный ресурс с определенным нами новым (нестандартным) типом LIFE и идентификатором, который в соответствии с системными соглашениями должен быть равен нулю. Кроме того, для представления прикладной программы и ее документов система использует пиктограммы, описанные в ресурсах *ICN#* и *ics#* (рис. 4), а также ресурс типа *FREF*, а необходимые связи и ссылки описываются в ресурсе типа *BNDL*.

В Life.r определяется строка с идентификатором 128. Текст, записанный в этот ресурс, содержит краткое описание программы и используется для передачи соответствующего сообщения *Менеджеру Помощи* (функции help, модуль LifeHelp).

Кроме отмеченных ресурсов, Life.r содержит также описание главного окна (тип *WIND*, идентификатор 128), два ресурса диалоговых запросов (alert box), тип *ALRT*, идентификаторы 128 и 129, и ресурсы типа *DITL*, описывающие диалог для этих запросов.

Главное окно, заданное координатами левого верхнего (22, 50) и правого нижнего (492, 318) углов, определено как окно неизменной величины, видимое (visible) при открытии, имеющее кнопку для закрывания (goAway) и заголовок «L I F E» (рис. 5).

Для ссылки на ресурсы диалоговых запросов *ALRT* в модуле констант *LifeConst* (файл *LifeConst.p*), исходный текст которого на языке Pascal приведен в листинге 2, определены две константы SAlertId и NAlertId со значениями, равными значениям идентификаторов этих ресурсов (то есть 128 и 129 соответственно). Кроме того, определена строковая константа creator, содержащая подпись программы, указанная также и в ресурсе *BNDL* листинга 1. По этой подписи операционная система будет отличать программу **Жизнь** и ее файлы данных (документы) от всех прочих файлов. Например, это нужно, чтобы изображать соответствующие пиктограммы в списке файлов (рис. 4), поэтому такая подпись должна быть уникальной и, чтобы не возникало путаницы, необходима ее регистрация для каждого программного продукта, распространяемого для работы на компьютерах фирмы «Apple». Однако, поскольку наша программа носит характер примера по программированию в исходных кодах и не предназначена для эксплуатации, мы не стали регистрировать подпись «LIFE», поэтому тем из читателей, кто захочет установить эту программу, нужно будет прежде убедиться, что на их компьютере нет какого-либо другого программного обеспечения с совпадающей подписью. Можно сменить подпись «LIFE» на любую другую по своему усмотрению или взять подпись «????» программ с отсутствующей индивидуальной подписью. В последнем случае пиктограмм **жизни** вы не увидите (рис. 6).

Программа жизни

Описав нужные нам ресурсы, можем перейти теперь к написанию самой программы. Для удобства мы разобьем ее на модули, как это уже было сделано для модуля констант LifeConst и упомянутого ранее модуля LifeHelp.

Сначала займемся главным (программным) модулем *Life* (файл *Life.p*, листинг 3). Здесь после заголовка программы зарезервированным словом «USES» открывается список системных модулей (естественно, не всех, а только нужных нам в этой программе) и остальных модулей программы **жизни** — для удобства их имена начинаются со слова Life.

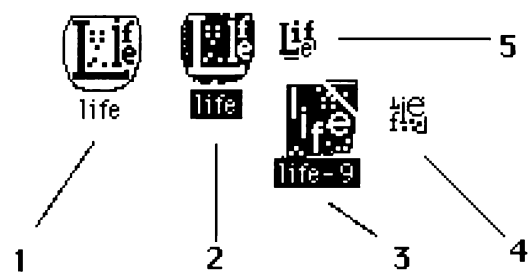


Рис. 4. Пиктограммы Жизни и прикладная программа:
1 — большая пиктограмма;
2 — отмеченная пиктограмма;
3 — отмеченный документ;
4 — малая пиктограмма документа;
5 — малая пиктограмма программы

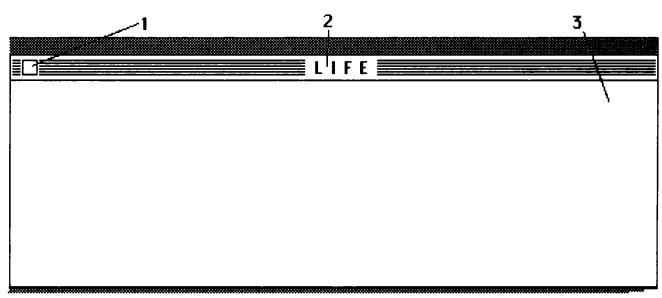


Рис. 5. Окно жизни:
1 — кнопка закрывания;
2 — заголовок;
3 — рабочая часть окна

Для удобства поиска процедур и функций в модулях каждый вызов мы будем сопровождать комментарием в соответствии с форматом:

```
{-ИмяМодуляГдеОпределенаПроцедураИлиФункция: текст комментария}
```

А вызовы стандартных процедур и функций комментариями:

```
{*ИмяМенеджера: текст комментария}
```

Чтобы не перегружать листинги, в ряде случаев явно избыточные комментарии будут опускаться.

В основном в нашем примере используются стандартные для Макинтоша структуры данных. Мы будем рассматривать их по мере надобности при описании модулей, использующих эти данные. Таким образом, модуль *LifeTypes* (файл *LifeTypes.p*, листинг 4) и модуль *LifeVar* (файл *LifeVar.p*, листинг 5) отдельно рассматриваться не будут.

Вернемся к программе Life. Первый оператор тела программы — вызов процедуры инициализации Init. Эта процедура определена в модуле *LifeInit*, который мы рассмотрим далее.

Следующий за вызовом Init оператор — установка значения переменной MaxGeneration, определяющей максимально возможное количество поколений:

```
MaxGeneration:=MaxInt;
```

Далее следует условный оператор:

```
if NoInitFiles then
```

Дело в том, что программа Life может быть запущена несколькими путями: двойным нажатием клавиши «мыши» (double click), когда ее курсор находится на пиктограмме этой программы, или двойным нажатием клавиши «мыши», когда курсор находится на каком-либо документе (файле), созданном этой программой. Третьим способом является следующий: пользователь устанавливает курсор на пиктограмму документа, нажимает клавишу «мыши» и, не отпуская клавиши, двигает захваченный документ, совмещая его пиктограмму с пиктограммой прикладной программы. В двух последних случаях документ должен быть загружен в прикладную программу с самого начала — в момент ее запуска. Функция NoInitFiles, определенная в модуле *LifeIO*, как раз и отслеживает эти случаи, проверяя, был ли указан какой-либо документ при запуске программы. Если никакого документа указано не было, функция принимает значение true, и происходит вызов функции help модуля *LifeHelp*. Результат работы этой функции булевского типа присваивается флагу hideHelp (спрятать help). В случае, если какой-либо документ был указан при запуске программы, флаг hideHelp сбрасывается. Далее вызывается процедура doInput — основная процедура редактора начальной конфигурации, модуля *LifeEditor*.

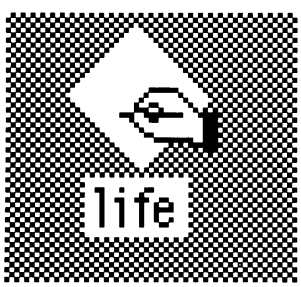


Рис. 6. Пиктограмма прикладной программы с подписью «????»

Отредактированная конфигурация записывается в файл данных под именем, хранящемся в строковой переменной OutName; если запись прошла успешно, о чем свидетельствует равенство кода ошибки, возвращаемого функцией «Записать файл» DoWriteFile модуля LifeIO, константе noErr, то имя файла запоминается в переменной StartState. Теперь все готово к тому, чтобы отобразить первое поколение и информацию о нем на экране в окне «L I F E», это достигается последующими вызовами трех процедур: NewGeneration, stamp и counts — модуля LifeMonitoring. Все начальные установки и отображения произведены — **жизнь** пошла.

Ход жизни

Ход **жизни** — смена поколений моделируется циклом

```
for generationNo :=2 to MaxGeneration do
 begin
```

Это последний (составной) оператор программы. Счетчик цикла (generationNo) является счетчиком поколений, а переменная MaxGeneration, как мы уже говорили, — максимальным количеством возможных поколений. Первый оператор тела цикла — вызов генератора нового поколения — процедура goOn модуля **LifeGenerator**. Эта процедура в соответствии с рассмотренными нами правилами **жизни** генерирует из живущего в данный момент поколения present будущее поколение — future. После чего новое поколение сменяет предыдущее:

```
present := future;
```

и отображается на экране вместе с сопутствующей ему информацией уже знакомым нам образом (вызовами процедур NewGeneration, stamp и counts). Следующий и последний составной оператор, вложенный в тело цикла, позволяет пользователю прервать программу; мы сейчас поговорим о нем подробнее, но прежде отметим, что цикл **жизни** на этом заканчивается.

```
 end {for}
end.{of Life}
```

Как прерывается жизнь

При описании ресурса окна мы предусмотрели стандартную кнопку для его закрывания (рис. 5): когда курсор находится на этой кнопке, нажатие клавиши «мыши» приводит к тому, что окно закрывается (исчезает с экрана), все другие открытые окна при этом остаются. Поскольку наша программа использует единственное окно, логично совместить с закрыванием окна выход из программы. Однако перед выходом нужно будет записать текущую конфигурацию в файл. Кроме того, полезно переспросить пользователя, действительно ли он хочет выйти из программы. Таким образом, намечается следующий порядок действий:

```
если поступила команда «закрыть окно»          (1)
то
  переспросить пользователя и если «да»         (2)
    то
```

```
закрыть окно                          (3)
записать конфигурацию в файл          (4)
сообщить пользователю имя этого файла (5)
прервать программу                    (6)
```

Строка (1) этого списка действий может быть реализована с помощью вызовов двух системных функций: функции GetNextEvent *Менеджера Событий (Event Manager)* и FindWindow *Менеджера Окон (Window Manager)*:

```
if GetNextEvent (...) then
if FindWindow (...) then
```

Поговорим сначала о событиях.

События на Маке (Менеджер Событий)

Согласно «Inside Macintosh», одним из ключевых элементов хорошо написанной для Макинтоша прикладной программы является ее «ориентированный на пользователя» дизайн. Это означает, что вместо предопределенной программистом последовательности шагов программа в большей мере управляется пользователем, то есть такими его действиями, как движение «мыши» и нажатие клавиш, порядок которых не может быть предписан заранее. Данный принцип реализуется использованием модели программирования управления событиями (an event-driven programming model).

Каждая прикладная программа может получать информацию о многих типах событий. Все возможные **события делятся на три категории**:

- события низкого уровня (low-level events);
- события операционной системы (operating-system events);
- события высокого уровня (high-level events).

Прикладная программа получает сообщение о событии путем вызова процедуры и функций *Менеджера Событий* (в нашем примере это функция GetNextEvent). В ответ на это *Менеджер Событий* возвращает очередное — в соответствии с приоритетом (event priority) — событие.

Информация о событии заносится в определяемую в прикладной программе переменную предопределенного в *Менеджере Событий* типа EventRecord:

```
TYPE EventRecord =
    RECORD
        what:       Integer;   {код события}
        message:    LongInt;   {сообщение события}
        when:       LongInt;   {время события в "тиках"}
        where:      Point;      {местоположение "мыши"}
        modifiers:  Integer;   {флаги модификатора}
    END;
```

Если мы не хотим, чтобы прикладная программа имела дело с событиями какого-либо типа, то их нужно фильтровать с помощью маски событий. Поскольку в данном случае нас интересуют только те, которые связаны с нажатием клавиши «мыши» (для закрывания окна), выбираем маску mDownMask.

Запишем теперь вызов функции GetNextEvent для нашей **жизни**. Эта функция определена в *Менеджере Событий* следующим образом :

```
FUNCTION GetNextEvent (eventMask: Integer;
                       VAR theEvent: EventRecord): Boolean;
```

где eventMask — маска событий; theEvent — запись события.

GetNextEvent возвращает FALSE в качестве результата, если событие «нулевое» или если оно перехватывается другим обработчиком события, в противном случае GetNextEvent возвращает TRUE.

Чтобы определить, указывал ли курсор на определенное место окна нашей программы, когда произошло событие нажатия клавиши «мыши» (нас интересует только клавиша для закрывания окна) или на какую-либо другую часть экрана (другое окно, линейку главного меню, Рабочий стол и так далее), нам нужно будет воспользоваться функцией *Менеджера Окон* FindWindow. Эта функция определена следующим образом:

```
FUNCTION FindWindow (thePoint: Point;
                     VAR theWindow: WindowPtr): Integer;
```

где thePoint — глобальные координаты точки; theWindow возвращается функцией и содержит указатель на окно, которому принадлежит данная точка или пустой указатель (nil) в противном случае.

Результат функции — целое число, указывающее на один из девяти регионов, которым соответствуют девять констант, определенные *Менеджером Окон*. В том числе:

```
inDesk - ни в одном из прочих регионов;
inMenuBar - линейка меню;
inGoAway - кнопка "Закрыть окно",
```

и так далее.

Теперь мы можем детализировать сформулированное нами прежде условие:

```
если поступила команда «закрыть окно»       (1)
```

Сделаем это, переписав его в окончательном виде (листинг 3):

```
if GetNextEvent (mDownMask, MainLoopEvent) then
    if FindWindow (MainLoopEvent.where, window) = inGoAway then
```

Здесь MainLoopEvent — переменная типа EventRecord (см. листинг 5 файла LifeVar.p); window — указатель на окно «L I F E» (мы еще вернемся к нему в дальнейшем).

Вспомним, что после условия (1) мы хотели

```
переспросить пользователя и если «да»     (2)
то ...
```

Для этого в файле ресурсов мы уже описали ресурсы диалоговых запросов типа *ALRT*. Подробнее мы расскажем о них в следующий раз. ■

Литература

1. **М. Трофимов. Pascal сегодня, вчера, завтра// Монитор. — 1993. — № 7-8.**
2. **Ч. Уэзерелл. Этюды для программистов. — М.: Мир, 1982.**

БЫСТРО, ПРОСТО, НАДЕЖНО

"ZyXEL многих заставил думать по-другому"
- из переписки в конференции RELKOM. POSTMASTERS

PC MAGAZINE EDITORS' CHOICE
June 29, 1993
ZyXEL U-1496 PLUS

JULY 1993 BYTE BEST VALUE HIGH-SPEED MODEM
ZyXEL U-1496E PLUS

Data Comm MAGAZINE Top Performer JULY 1993
ZyXEL U-1496 PLUS

COMPUTER SHOPPER
July 1993, ZyXEL U-1496E PLUS
ZyXEL U-1496 PLUS
Top 10 List

JULY 1993 BYTE BEST TWO-WAY ALL-AROUND MODEM
ZyXEL U-1496E

PC PROFESSIONELL EMPFEHLUNG DER REDAKTION
February 1993
ZyXEL U-1496E PLUS

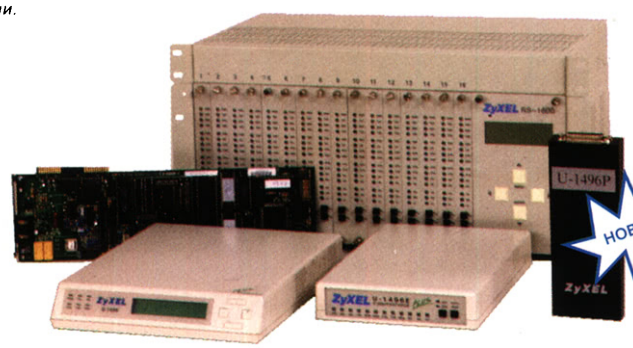
Некоторые считают, что ZyXEL - самый хороший из быстрых модемов. Другие считают, что ZyXEL - самый быстрый из хороших модемов. Мы согласны и с тем, и с другим.

Вы спросите почему? Причины очевидны: Главный критерий оценки модема - надежность его работы. Именно поэтому ZyXEL стал стандартом de-facto для высокоскоростных модемов в России. Банки, торговые дома, транспортные компании, предприятия топливно-энергетического комплекса, сети передачи данных - все, кто знает цену своей информации, доверяют ее модемам ZyXEL.

19200 для модемов ZyXEL - это не просто внушительное число, написанное на коробке, а реальная физическая скорость, которая дает вам экономию времени и денег.

Более того, удивительно широкий набор дополнительных функций модема позволит вам сразу решить и другие коммуникационные проблемы: прием и рассылка факсов, организация речевой почты (interactive voice mail) и комплексных информационных систем.

Широкий спектр моделей серии U-1496 полностью соответствует многообразию задач, которые стоят перед вами. Семейство модемов в промышленном исполнении и NMS позволят осуществить контроль и управление сетью из сотен модемов одновременно из одного места. Для тех, кто не может позволить себе быть без связи в дороге, новый портативный модем U-1496P станет незаменимым дополнением к переносному компьютеру и сотовому телефону.



Все модемы ZyXEL совместимы с большинством других модемов и могут работать в синхронном или асинхронном режиме с такими программными средами, как DOS™, Windows®, OS/2®, Macintosh®, UNIX®, NeXT®, Amiga™, Atari™. В синхронном режиме с использованием команд V.25bis. ZyXEL совместим с системами AS/400® и RS/6000® фирмы IBM.

Основные характеристики модемов ZyXEL:

- *Высокая скорость 19200 bps*
- *Сверхнадежный режим ZyCELL*
- *V.32bis/V.32, V22bis/V.22, BELL 212A*
- *V.17 - 14400 bps, ELA Class 2, G3 Fax, S/R*
- *V.42/V42.bis (+ Selective Reject), MNP® 3/4/5*
- *Цифровая запись/воспроизведение звука*
- *АОН, распознавание условных звонков*
- *Дистанционное конфигурирование*
- *Динамический выбор рабочей скорости*
- *Защита от НСД: пароли, обратный звонок*
- *Адаптация к абонентской линии*
- *2/4 проводная коммутируемая/выделенная линия*
- *Автоматическое распознавание режима Data/Fax/Voice*
- *Перепрограммируемое ПЗУ —обновление микропрограммы*

yes
It runs with NetWare

Сертификат Министерства Связи России

Приобретая модемы ZyXEL у авторизованных дилеров МКЦ "Вариант" вы, кроме всего этого, получите оборудование, произведенное специально для России, имеющее сертификат Министерства Связи, необходимые дополнительные программы, 2 года гарантии со склада, бесплатное обновление микропрограммы, профессиональную поддержку квалифицированного технического персонала и доступ к BBS.

ZyXEL

Официальный дистрибьютор
АО МКЦ "Вариант"
117279, Москва, Островитянова 37а
Тел. (095) 420 2519
Факс.(095) 420 5311

Круглосуточно
Информация (095) 932 8510
Электронная почта
max@mechta.msk.su, 2:5020/22@fidonet
WHITE BEAR BBS - (095) 932 8465

Листинг 1.
Исходный файл ресурсов Life.r

```
resource 'WIND' (128, "main window", purgeable) {
    {50, 22, 318, 492},
    noGrowDocProc,
    visible,
    goAway,
    0x0,
    "L I F E"
};

resource 'ALRT' (128, purgeable) {
    {228, 178, 304, 463},
    128,
    {    /* array: 4 elements */
        /* [1] */
        OK, visible, sound1,
        /* [2] */
        OK, visible, sound1,
        /* [3] */
        OK, visible, sound1,
        /* [4] */
        OK, visible, sound1
    }
};

resource 'ALRT' (129, purgeable) {
    {106, 84, 224, 407},
    129,
    {    /* array: 4 elements */
        /* [1] */
        OK, visible, sound1,
        /* [2] */
        OK, visible, sound1,
        /* [3] */
        OK, visible, sound1,
        /* [4] */
        OK, visible, sound1
    }
};

resource 'DITL' (128, purgeable) {
    {    /* array DITLarray: 3 elements */
        /* [1] */
        {50, 200, 70, 258},
        Button {
            enabled,
            "Quit"
        },
        /* [2] */
        {50, 110, 70, 185},
        Button {
            enabled,
            "Continue"
        },
        /* [3] */
        {10, 70, 32, 221},
        StaticText {
            enabled,
            "Quit from this Life?"
        }
    }
};

resource 'DITL' (129) {
    {    /* array DITLarray: 2 elements */
        /* [1] */
        {81, 255, 101, 313},
        Button {
            enabled,
            "Ok"
        },
        /* [2] */
        {16, 76, 71, 317},
        StaticText {
            enabled,
            "The life saved in file(s) :\n
            "        ^0 \n                ^1 ("
            "final state) ."
        }
    }
};

resource 'STR ' (128) {
    "The Life game . This is the torus where "
    "the Life of cell's population has been p"
    "roceeded.To insert initial cells click m"
    "ouse button  in torus area, at the end c"
    "lick close box of the LIFE window. Click"
    " once more the box to exit from the game"
    "."
};

resource 'BNDL' (128) {
    'LIFE',
    0,
    {    /* array TypeArray: 2 elements */
        /* [1] */
        'FREF',
        {    /* array IDArray: 2 elements */
            /* [1] */
            0, 128,
            /* [2] */
            1, 129
        },
        /* [2] */
        'ICN#',
        {    /* array IDArray: 2 elements */
            /* [1] */
            0, 128,
            /* [2] */
            1, 129
        }
    }
};

resource 'FREF' (128) {
    'APPL',
    0,
    ""
};
```

```
    resource 'FREF' (129) {
        'TEXT',
    1,
    ""
};

resource 'ICN#' (128) {
    {    /* array: 2 elements */
        /* [1] */
        $"1FFF FFFC 1000 0602 2FE0 061A 2FE0 003A"
        $"4380 0E61 4380 0E61 8399 86F1 8399 8661"
        $"8380 0661 8399 8661 8399 86F1 8380 0601"
        $"8386 0631 8386 0679 8380 06CD 8380 06FD"
        $"8380 06F9 838C 06C1 838C 0679 83B0 CF3A"
        $"43B0 CF02 4380 0006 4380 0064 2FFF FFE4"
        $"2FFF FFEC 2FFF FFE8 1000 0018 0800 0020"
        $"0C00 00C0 0780 0780 00FF FC",
        /* [2] */
        $"018B 376C 0FFF B9FE 3FFF FFFE 3FFF FFFE"
        $"7FFF FFFF 7FFF FFFF DFFF FFFF 7FFF FFFE"
        $"5FFF FFFE FFFF FFFF 7FFF FFFF 5FFF FFFF"
        $"FFFF FFFF 7FFF FFFF 5FFF FFFF 7FFF FFFF"
        $"FFFF FFFF 6FFF FFFF 7FFF FFFF FFFF FFFE"
        $"7FFF FFFE 7FFF FFFA 7FFF FFF8 3FFF FFFC"
        $"3FFF FFFC 3FFF FFF8 1FFF FFF8 0BFF FFE0"
        $"02FC FF40 0678 E700 00FB CC"
    }
};

resource 'ICN#' (129) {
    {    /* array: 2 elements */
        /* [1] */
        $"FFFF E000 A000 7000 A000 3800 A601 9E00"
        $"A601 8E00 A600 1F30 A600 1BB0 A600 67C0"
        $"A600 66E0 A600 0070 A600 6638 A660 661C"
        $"A660 000E A600 01C7 A660 07F3 A660 0631"
        $"A060 0C19 A061 CFF9 A063 DFF9 A063 1C01"
        $"A067 C631 A067 C7F1 A063 01C1 A063 0001"
        $"A003 0019 A003 1819 A003 1801 A303 01B1"
        $"A303 01B5 ACC3 0407 ACC0 43FF A01F FFFF",
        /* [2] */
        $"FFFF FFFF 8000 0001 BFFF FFFF BFFF FFFF"
        $"BFFF FFFF BFFF FFFF BFFF FFFF BFFF FFFF"
        $"BFFF FFFF BFFF FFFF BFFF FFFF BFFF FFFF"
        $"BFFF FFFF BFFF FFFF BFFF FFFF BFFF FFFF"
        $"BFFF FFFF BFFF FFFF BFFF FFFF BFFF FFFF"
        $"BFFF FFFF BFFF FFFF BFFF FFFD BFFF FFFD"
        $"BFFF FFFD BFFF FFFD BFFF FFFD BFFF FFFD"
        $"BFFF FFFD BFFF FFFD BFFF FC01 FFFF FFFF"
    }
};

resource 'ics#' (129) {
    {    /* array: 2 elements */
        /* [1] */
        $"0004 223A 2045 2282 2AFE 7A80 0280 1A42"
        $"223C 2201 7261 276D 200D 2331 7331 000F",
        /* [2] */
        $"FFF8 FFFE FFFE FFFF FFFF FFFF FFFF FFFF"
        $"FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF"
    }
};

resource 'ics#' (128) {
    {    /* array: 2 elements */
        /* [1] */
        $"0186 798C 300C 339E 338C 318C 318C 319E"
        $"33C1 3019 3025 30BD 7FA1 7FA5 0019 FF86",
        /* [2] */
        $"7FFE FFFE FFFE FFFF FFFF FFFF FFFF FFFF"
        $"FFFF FFFF FFFF FFFF 7FFE 7FFE 7FFE 1FF0"
    }
};

data 'LIFE' (0, "Owner resource") {
    $"134D 6963 6861 656C 2049 2E20 5472 6F66"   /*Michael I. Trof
*/
    $"696D 6F76"                                 /* imov */
};
```

■

Листинг 2. Константы ЖИЗНИ (файл LifeConst.p)

```
UNIT LifeConst;
 {глобальные константы}

 INTERFACE

  CONST
   fieldHeight = 30; {размеры поля: - высота}
    fieldWidth = 50;  {            - ширина}
    SAlertId = 128; {идентификатор ресурса запроса остановки}
   NAlertId = 129; {идентификатор ресурса запроса замечания}
    creator = 'LIFE'; {подпись программы}

 IMPLEMENTATION
END. {LifeConst}
```

■

Листинг 3. Программа ЖИЗНЬ (файл Life.p)

```
PROGRAM life;
 {The Life game}

{*************************************************************}
{* Игра ЖИЗНЬ.                                               *}
{* Пример программирования компьютера Макинтош               *}
{* (C) Copyright by Michael I. Trofimov, 1994.               *}
{* Все авторские права сохраняются за автором.
*}
{* Допускается некоммерческое использование для целей образования *}
{*************************************************************}


 USES

    MemTypes,QuickDraw,OSIntf,ToolIntf,PackIntf,Traps,Balloons,
    LifeConst,LifeTypes,LifeVar,LifeEditor,LifeGenerator,
   LifeMonitoring, LifeInit,LifeIO,LifeHelp;

 VAR
  generationNo: integer; {Счетчик цикла  должен быть локальной
                                         переменной}

 BEGIN { life }
   init; {-LifeInit: инициализация}
```

```
  MaxGeneration:=MaxInt; {установка максимального
                 числа поколений}
 IF NoInitFiles {-LifeIO: Открытие документов при запуске
               программы}
  THEN
    hideHelp:=help {-LifeHelp: установить помощь}
  ELSE
    BEGIN
  hideHelp:=false; {спрятать окна помощи}
  stamp(present); {-LifeMonitoring: отображение на экран
                                  одного поколения}
    END;
 doInput(present);       {-LifeEditor: ввод начальной конфигурации}
IF DoWriteFile=noErr  {-LifeIO: запись начальной конфигурации}
THEN
  StartState:=OutName; {запомнить имя документа}
   NewGeneration(1);     {-LifeMonitoring: изменить заголовок
                         окна ЖИЗНИ}
   stamp(present); {-LifeMonitoring: отображение на экран
                   поколения}
  counts(1); {-LifeMonitoring: отображение на экран
            информации ЖИЗНИ}

FOR generationNo:=2 TO MaxGeneration DO
  BEGIN
   goOn(present,future); {-LifeGenerator: генерация
                          нового поколения}
    present:=future; {смена поколений}
    stamp(present); {-LifeMonitoring: отображение на экран
                    нового поколения}
   NewGeneration(generationNo); {-LifeMonitoring:
                        изменить заголовок  окна ЖИЗНИ}
   counts(generationNo); {-LifeMonitoring: отображение
                          на экран  информации ЖИЗНИ}
  IF GetNextEvent(mDownMask,MainLoopEvent) {*Event: получить
                                 информацию о событии}
  THEN
   IF FindWindow(MainLoopEvent.where,window)=inGoAway
     {*Window: найти  часть окна,  указанную пользователем}
   THEN
     IF StopAlert(SAlertId,NIL)=1 {*Dialog: запрос выхода
                                   из ЖИЗНИ}
      THEN
      BEGIN
        CloseWindow(window); {*Window: закрыть окно ЖИЗНИ}
       IF (DoWriteFile=noErr) {-LifeIO: записать конечную
                               конфигурацию в файл}
           OR (StartState'')
       THEN
      BEGIN
  {синтезировать сообщение об имени документа для запроса}
        IF StartStateOutName
         THEN
         BEGIN
           StartState:=concat(StartState, ' (start state)');
            ParamText(StartState,OutName,'','');{*Dialog}
         END
         ELSE
            ParamText('',OutName,'','');
           IF NoteAlert(NAlertId,NIL)=1 {*Dialog: запрос
                                  сообщение:имя документа}
             THEN {ничего не делать} ;

           END;

         halt; {остановить программу}
         END; {if StopAlert}

    END {for}
 END. {of Life}
```

■

Листинг 4. Типы ЖИЗНИ (файл LifeTypes.p)

```
UNIT LifeTypes;
 {глобальные типы}

 INTERFACE

 USES
    MemTypes,QuickDraw,OSIntf,ToolIntf,PackIntf,Traps,
    LifeConst;

 TYPE
   field = ARRAY [1..fieldWidth,1..fieldHeight] OF boolean;
                                                { Поле ЖИЗНИ}

 IMPLEMENTATION
END. {LifeTypes}
```

■

Листинг 5. Переменные ЖИЗНИ (файл LifeVar.p)

```
UNIT LifeVar;
 {глобальные переменные}

 INTERFACE

  USES
    MemTypes,QuickDraw,OSIntf,ToolIntf,PackIntf,Traps,
    LifeConst,LifeTypes;

  VAR

   present,future: field; {массивы настоящего и будущего поколений}
   torus,CurrentRect,incRect: rect; {прямоугольники для отрисовки
                                в окне ЖИЗНИ}
    hideHelp: boolean;          {флаг помощи}
   MaxGeneration,               {максимальное число поколений}
   increasing: integer;         {информационный параметр "прирост"}
   window: windowPtr;           {окно ЖИЗНИ}
   gDone: boolean;              {флаг цикла событий}
    MainLoopEvent: EventRecord; {запись события}
    StartState,OutName: Str255; {имена документов}
    StartPoint,CurrentPoint,increasePoint: point; {координаты
                    точек  для расположения информационного табло}
    StartCount,CurrentCount,numberOfCells: integer;
                                {информационные  параметры}
    StepForward: ARRAY [boolean] OF integer; {единичные сдвиги
                                 для подсчета соседей}

 IMPLEMENTATION
END. {LifeVar}
```

■

Представляем вам Независимую Тестовую Лабораторию (НТЛ)

НТЛ проводит тестирование и анализ программных продуктов, компьютеров, аппаратуры для компьютеров, периферийных устройств, сетей, вычислительных комплексов. Результаты публикуются в печати.

НТЛ вырабатывает свод критериев качества и выдает разработчикам, производителям и другим заинтересованным лицам и организациям сертификаты о соответствии их продукции своду критериев качества **НТЛ**.

Независимая Тестовая Лаборатория журнала «Монитор» приглашает вас принять участие в тестировании аппаратных и программных средств российских и зарубежных производителей. Программа тестирования предварительно согласуется с заказчиками.

Пожалуйста, заполните отрезную карточку, отметив заинтересовавшие вас виды тестирования. Также можете вписать свои предложения в свободную строку.

Заполненную карточку вышлите **по адресу**:
117419 Москва, а/я 765 или **по факсу** ***(095)238-9911.***

Стоимость работ включает **50 экземпляров бюллетеня** в твердом переплете с результатами тестирования и **публикацию в журнале «Монитор»** объемом до 4 полос.

Фирма _____

Адрес _____

_____

Телефон (___) _____

Факс (___) _____

Ф.И.О. _____

№ п/п	Тематика тестирования	Сроки проведения	Стоимость работ (без НДС и СН), $США	Готовы принять участие
1	Тестирование лазерных принтеров ценой до 1000 долларов США	март-апрель 1995г.	800	
2	Тестирование графических рабочих станций	апрель-май 1995г.	2000	
3	Тестирование программ для работы мультимедиа-приложений	май-июнь 1995г.	600	
4	Тестирование высокопроизводительных жестких дисков	июнь-июль 1995г.	700	
5				

Boston PC *Компьютеры из США*

Настольные

Pentium-90/16/1G/SVGA.28	$3400
486DX2-66/8/1G/SVGA.28	$1960
486DX2-66/4/420/SVGA.28	$1290
486/7SLC2-50/4/260	$1000

Notebook

AMBRA 486DX2-50/8/200/CAM	$2800
SHARP 486DX-33/4/200/CAM	$2350
Dauphine 486SLC-25/6/40/f-m/pen	$830
Dauphine 486SLC-25/4/40/f-m/pen	$750

Факс-модемы

US Robotics 28800 ext.	$280
US Robotics 28800 int.	$230
US Robotics 14400 ext.	$140
US Robotics 14400 European	$165
US Robotics 14400 int.	$125
US Robotics PCMCIA 14400	$280
Zyxel 1496E	$315
Zyxel 1496E+	$415
GVC 14400 int.	$95
BOCA 28800 int.	$190
IBM PCMCIA 14400	$180
IBM PCMCIA 9600	$100

Звуковые карты

Sound Galaxy 16 Basic	$110
Sound Blaster 16 Value Edition	$115
Sound Blaster 15 MCD w/ASP	$195
Sound Blaster 16 SCSI w/ASP	$230
Sound Blaster 32 AWE VE	$250

CD-ROM Drive

Panasonic 562B	$155
Mitsumi 2-x скор. IDE	$150
NEC 510 3-x скор. SCSI	$290
Mitsumi 4-x скор. IDE	$230
Toshiba 4-x скор. SCSI	$390

Игры и обучающие программы на CD по ДЕЙСТВИТЕЛЬНО самым низким ценам в Москве !

Mad Dog McCree	$12	Lawmmower Man	$16	Microsoft Office	$12	Библиотека предпринимателя	$200
Mad Dog McCree 2	$20	Goblin III Quest	$17	Microsoft Encarta	$12	Сборник программ для Windows	$80
7-th Guest	$20	Legend of Kurandia	$23	Learn to speak English	$12	MIDI Workshop Professional	$14
Myst	$12	Tornado	$12	World Atlas	$14	Animals	$14
Doom Fever	$12	Space Pirates	$12				
Mega Race	$13	Crime Patrol	$12				

А также более 100 других наименований

м. "Беговая". 1-й Хорошевский пр. д. 12 корп. 1 подъезд 3
Тел 946-01-11, 946-07-06

ВВЦ павильон "Центральный" 1 этаж

м. "Свиблово" т. 189-27-65

МонитОр 2.95

Аспект

PC и MACINTOSH в графических операционных средах

Программы

MathCAD - пять с плюсом

Образование

Компьютерные классы

Мультимедиа

Мультимедиа в Windows

PC и Макинтош в графических операционных средах

А.Мурадян

В статье приведены результаты тестирования Независимой Тестовой Лабораторией рабочих станций, предназначенных для работы в графических операционных средах.

Вступление: выбор объектов тестирования

Тесная связь между прогрессом в программной области и развитием аппаратных средств приводит к тому, что вслед за резким скачком в развитии «железа» происходит скачок в области разработки программ и наоборот. Практически всю эволюцию компьютерной техники можно разбить на «полосы» развития, где в качестве «ведущих» и «ведомых» попеременно выступали то «hard», то «soft».

Появление 486-го процессора, высокоскоростных жестких дисков позволило говорить о том, что PC-платформы из разряда пишущих машинок, калькуляторов и игрушек с цветным экраном переходят в разряд машин для серьезной работы. Правда, в отношении программного обеспечения для PC сохранялась определенная доля скептицизма. Сравнение возможностей по графическим и мультимедийным работам на MAC, NeXT, выполнением расчетов на SUN со скоростью и удобством тех же операций на PC было заведомо не в пользу последних. Однако вскоре последовал выпуск Windows 3.0, а затем и 3.1 и настоящий взрыв перевода программ под Windows, появились возможность работы с 24-битовыми (а затем и 32-битовыми) изображениями и поддержка мультимедиа, более или менее совместимый интерфейс прикладных программ, приемлемая скорость в обработке данных, новые высокоскоростные локальные шины VESA и PCI. Все это заставило пользователей других платформ рассматривать PC как возможную альтернативу, исходя из соотношений цены, количества наработанного программного обеспечения, производительности и удобства в использовании.

Если в хаотичном мире PC царила непредсказуемость, а производителей и разработчиков подстегивала жесточайшая «внутриплатформенная» конкуренция, то в стане компьютеров Макинтош сложилась совершенно противоположная ситуация: почти полный захват рынка компьютеров для образования и настольных издательств, наилучшая и наиболее удобная на тот момент графическая операционная среда, отсутствие конкурентов, так как «маки» выпускает только сама фирма «Apple».

Исходя из сказанного, нам было интересно провести тестирование и последующий сравнительный анализ рабочих станций среднего класса, предназначенных для работы в графических операционных средах (не путать с графическими рабочи-

---

Настоящая статья подготовлена по результатам тестирования, проведенного в Независимой Тестовой Лаборатории, и представляет собой краткое изложение полученных в ходе тестирования результатов. Полное изложение результатов тестирования содержится в бюллетене НТЛ, который можно приобрести в редакции журнала «Монитор».

Контактные телефоны НТЛ: (095) 284-57-28, (095) 238-99-33.

ми станциями, здесь совершенно другие, значительно более серьезные требования к аппаратуре).

Участники

Критерии отбора участников тестирования были следующими. Чтобы сократить процесс проведения тестовых испытаний и в то же время не урезать саму программу тестирования, решено было ограничиться четырьмя участниками. Фирма «Apple» предоставила для тестирования два компьютера: Mac Quadra 800 и Power Mac 8100, фирма «CompuLink» — компьютер CLR и фирма «Классика» — Gateway2000. Выбор участников тестирования был не случаен. Интересно было сравнить, как отечественные компьютеры производства фирмы «CompuLink» будут смотреться на фоне одного из лидеров компьютерного рынка США Gateway и компьютеров компании «Apple», изначально ориентированных на работу в графической среде.

Требования к аппаратному обеспечению для PC

1. Минимум — процессор 486DX2 с частотой 66 МГц и внешним кэшем в 256 кб.
2. Размер ОП не менее 16 Мб.
3. Размер видеопамяти не менее 1 Мб.
4. Жесткий диск емкостью не менее 300 Мб.
5. Накопитель на CD с удвоенной скоростью.

Цель тестирования

Тестирование, проводимое Независимой Тестовой Лабораторией, должно было дать ответ на вопрос: насколько обоснованны сообщения фирм о мощи своих компьютеров. Отметим однако, что полученные результаты отражают только то, насколько быстро и хорошо работают конкретные экземпляры компьютеров на конкретных экземплярах программ в конкретных операционных средах.

Программа тестирования

Составляя методику тестирования, трудно было удержаться от искушения перепробовать весь спектр программного обеспечения, от игрушек до промышленных систем, однако мы остановились на некоем среднем наборе тестов, позволяющих получить достаточно полное представление о производительности различных подсистем компьютера при работе в графических операционных средах.

Программа тестирования для PC выглядела следующим образом.

В качестве основы для тестирования были выбраны графические операционные среды Windows 3.1, OS/2 3.0 Warp и X-Windows (Linux — некоммерческая версии UNIX), подход к реализации графического интерфейса у которых существенно отличается.

Для Windows использовались следующие тестовые задачи.

1. Работа с растровой графикой. Растровое изображение переводилось в формат RGB True Color, затем в CMYK True Color с последующим цветоделением и выводом на экран в пакете Photo Styler 2.0.
2. Вывод векторной графики. Изображение в формате Corel Draw! выводилось на экран в программе Corel Draw! 5.0 путем чтения файла с жесткого диска.
3. Работа с базами данных. Файл в формате dBase IV загружался с диска с импортом в формат XLS и последующей сортировкой одного поля по алфавиту и выводом на экран. В качестве тестирующей программы был выбран Microsoft Excel 5.0.
4. Компиляция контрольного примера на языке C. Компиляция проводилась с помощью компилятора Microsoft Visual C++ 1.5 Professional, в двух вариантах: с подключением библиотек, находящихся на жестком диске и на CD.
5. Работа с накопителем на компакт-дисках. Проводилось сканирование CD-диска при помощи Microsoft AntiVirus.
6. Работа с текстами. Производились поиск и замена определенной комбинации букв в текстовом файле в редакторе Word for Windows 6.0.

Для OS/2 использовался иной набор тестовых задач.

1. Загрузка OS/2 для суммарной оценки быстродействия всех подсистем компьютера.
2. Для оценки быстродействия накопителя на компакт-дисках и эффективности работы OS/2 Windows, проводилось сканирование того же компакт-диска, который использовался при тестировании в Windows, при помощи Microsoft AntiVirus.
3. Работа с текстами. Глобальная замена шрифта в текстовом процессоре программы IBM Works.
4. Создание трехмерного ландшафта (обсчет и генерация теней) и вывод его на экран.

Для X-Windows использовались следующие тестовые задачи.

1. Суммарная оценка быстродействия: запуск X-Windows.
2. Загрузка с компакт-диска растрового изображения в программе XV 3.0.
3. Фильтрация того же изображения в той же программе.
4. Выполнение расчета и построение фрактала в программе Fractinit.

Кроме оценки производительности компьютера при помощи указанных тестов, оценивались также удобство в работе, корректность работы видеосистемы в различных графических режимах, возможности расширения системы.

Программа тестирования компьютеров Макинтош была практически идентична программе тестирования для Windows, но с небольшими изменениями.

1. Тест с растровой графикой проводился на программе Adobe Photoshop (для Quadra — коммерческая версия 3.0, для PowerMac — бета-версия этой же программы в кодах Power PC).
2. Ввиду невозможности использования напрямую векторного формата cdr в программах на Mac, тестовый пример был конвертирован в векторный же формат для программы Mac Draw Pro.
3. Так как перенос контрольного примера на языке C был практически невозможен, контрольный пример был заменен на близкий по объему, а в качестве компилятора мы использовали Think C.
4. Из-за того что формат текста, который мы использовали, был «чужим» для программ Mac, мы выполнили тестовую задачу на двух текстовых процессорах Mac Write Pro 4.0 и Microsoft Word Mac 2.0.
5. Сканирование компакт-диска выполнялось при помощи программы Desinfect.

Описание компьютеров-участников и результаты тестирования

«CompuLink» — CLR Computer

CLR Computer выглядит, на первый взгляд, достаточно обыденно, однако комплектующие, которыми начинен компьютер, заурядными назвать трудно. Интригу в процессе тестирования усилил тот факт, что НПО «CompuLink» выставило не вылизанную специально для тестирования, а некую экспериментальную модель, собранную техническим отделом фирмы для проверки ее характеристик в перспективных графических средах. В качестве основной платы специалисты фирмы остановили свой выбор на плате с набором чипов SIS и с локальной шиной VESA. Из слотов расширения, а их 7 (4 — 16-разрядных и 3 — VESA), под контроллеры занято 4, так что свободными остаются всего один разъем VESA и два 16-разрядных. Разъемов для установки модулей памяти типа SIMM — восемь, так что возможности по расширению оперативной памяти можно оценить как хорошие. Контроллер накопителей на жестком и гибком дисках — фирмы «Tekram» с памятью в 4 мегабайта. В качестве других компонентов внешней памяти в компьютере используются высококачественные изделия известных фирм: высокопроизводительный жесткий диск емкостью 540 мегабайтов фирмы «Maxtor» и очень неплохой 3-дюймовый дисковод фирмы «Panasonic». Что касается накопителя на компакт-дисках, то использование накопителя фирмы NEC и SCSI-2 адаптера Pro Audio в комплекте, поставляемом фирмой «Media Vision», можно считать удачным ввиду надежности в работе.

В качестве контроллера портов первоначально была установлена мультикарта фирмы «Gold Star», однако в процессе установки X-Windows произошел выход из строя не только мультикарты, но и основной платы, по причине несовместимости контроллера дисков с мультикартой, а самой мультикарты с Linux. Кроме того, в процессе работы было выяснено, что у указанной мультикарты был сильно «урезан» второй последовательный порт. В результате тестирования «CompuLink» полностью отказалась от контроллеров «Gold Star».

Графическая подсистема была представлена сначала акселераторной платой на чипе Tseng ET4000 с 2 мегабайтами памяти стандарта VESA, однако в процессе тестирования к ней добавились еще два адаптера: адаптер с 2 мегабайтами памяти и микросхемой S3 805 и 1 Мб контроллер на микросхеме Trident 9400. Монитор фирмы «Acer» с диагональю в 15 дюймов в целом неплохо справлялся с отображением информации, однако при режиме 1024×768 по краям появлялись искажения, которые оставались даже после регулировки. Клавиатура фирмы BTC и мышь «Made in Taiwan» со своими функциями справлялись без каких-либо проблем. Гарантия в два года и ремонт в течение 24 часов вызывают только положительные эмоции. Цена за этот комплект составляет 3300 долларов.

Самым интересным результатом тестирования было то, что практически во всех тестах (исключая обработку векторной графики и сканирование компакт-диска) CLR оказался почти вровень с Mac Quadra, а в тесте на сортировку по алфавиту опередил даже Power Mac.

Результаты «внутрифирменного» сравнения трех вариантов компьютера CLR дают обильную пищу для размышлений.

Компьютер с видеоконтроллером Tseng ET4000, оказавшийся очень быстрым при выполнении практически всех тестовых задач в Windows (в тесте на конвертирование RGB CMYK он был

быстрее, чем компьютер Gateway2000, показавший наилучшие результаты в 14 тестах из 18 при сравнении PC), исключая тест на поиск и замену текста, где указанная операция происходила в течение одной минуты (у других PC результат в диапазоне от 30 до 12 секунд), показал посредственные результаты при работе в X-Windows и OS/2 (наихудшие результаты в 2 тестах из 8). Возможно, это связано не с медленной работой самого контроллера ET4000, а с неудачной реализацией Video-BIOS фирмы «Quadtel». Кстати, контроллер с чипом S3 имел BIOS той же фирмы.

Второй вариант компьютера CLR с видеоконтроллером с чипом S3 оказался наиболее медленным из всех: абсолютно наихудший результат, показанный в Windows (8 тестов из 10) и X-Windows (4 из 4). Только в загрузке OS/2 он опередил всех.

Несмотря на то, что видеоадаптеры с ET4000 и S3 были укомплектованы двумя мегабайтами памяти, а Trident 9400 всего одним, компьютер, в котором был установлен видеоконтроллер Trident, показал очень неплохие результаты и оказался последним всего лишь в двух тестах из восемнадцати.

Из трех вариантов компьютеров «CompuLink» наилучшее впечатление осталось у нас от компьютера с видеоадаптером Trident. В целом, компьютеры «CompuLink» производят неплохое впечатление, однако для того чтобы довести их до подобающего уровня, специалистам из «CompuLink» предстоит немало поработать. Однако, учитывая тот фактор, что фирма не побоялась выставить для сравнения с более именитыми «Apple» и «Gateway» экспериментальную модель, которая выглядела на их фоне не так уж слабо, можно только приветствовать желание специалистов фирмы доводить свой продукт до качественного уровня не путем экспериментов на покупателях, а с помощью беспристрастного и достаточно жесткого сравнения с грандами компьютерного рынка.

«Apple Computer»—Quadra 800 и PowerMac 8100

Quadra 800

Этот компьютер с появлением компьютеров серии PowerMac несколько отошел в тень, однако по сей день не утратил своего значения как неплохая рабочая станция. Компьютер, предоставленный для тестирования, был укомплектован 250 Мб жестким диском, 8 Мб оперативной памяти, прекрасным 16-дюймовым монитором и внешним накопителем на компакт-дисках. Следствием тщательного подхода фирмы «Apple» к подбору комплектующих для своих компьютеров, а также отсутствия разнобоя в аппаратных и программных средствах, имеющегося на рынке PC, явилось то, что надежность, продуманность в комплектации и удобство «Маков» в эксплуатации можно без преувеличения назвать отличными.

Возможности расширения системы можно оценить как очень хорошие: трех слотов расширения вполне хватит для наращивания мощностей компьютера (то же самое относится и к Power Mac), так как встроенные сетевая поддержка и возможности мультимедиа изначально интегрированы в систему. В связи с тем, что комплектация Mac является типовой для всех компьютеров одной серии, не будем останавливаться более подробно на перечислении комплектующих ни для Quadra, ни для PowerMac, а сразу перейдем к результатам тестирования.

Все тестовые задачи прошли на компьютере без проблем, однако если в обработке графики были показаны неплохие результаты (особенно это касалось обработки векторной графики, где преимущество перед PC-компьютерами было подавляющим), то в обработке текстовой информации и в работе с базой данных были получены невысокие результаты (здесь PC-платформы взяли убедительный реванш, даже самая медленная машина выполнила операции по сортировке баз данных на порядок быстрее, в обработке текста же преимущество было не столь заметным).

В обработке растровой графики очень быстро выполнялось цветоделение по CMYK (всего около 2 секунд), что в общем-то и подтверждает тот факт, что «Маки» как рыба в воде чувствуют себя при выполнении графических работ.

При выполнении теста с векторной графикой для вывода на экран достаточно большого изображения, состоящего из 500 объектов, Quadra затратила всего 8 секунд, что, честно говоря, повергло в уныние тех, кто считал, что PC если и проиграют, то не со столь заметным разрывом.

Время, затраченное на сканирование компакт-диска, составило около 10 минут, что на 3 минуты меньше, чем потребовалось PC-компьютерам. Вероятно, это связано с тем, что сама программа тратила меньше времени на открытие двоичных файлов «неродного» формата.

Текстовую информацию Quadra обрабатывала мучительно долго. На в общем-то средний файл (всего 6500 замен) она затратила более 4 минут в программе Word Mac и 3,5 минуты в редакторе Mac Write

Pro, причем текст был конвертирован в формат, «родной» для программы.

Компиляция тестового примера заняла более двух минут. Это можно считать отнюдь не впечатляющим результатом, так как, даже подключая библиотеки с компакт-диска, PC выполняли эту операцию быстрее.

Наиболее долго выполнялся тест на обработку баз данных. Если импорт файла в формате DBF выполнялся за 2 минуты (на PC порядка 22 секунд), то процесс сортировки 12,5 тысяч записей тянулся 3,5 минуты, а на самом медленном PC эта операция заняла 7 секунд.

PowerMac 8100

Наиболее мощная модель в семействе компьютеров фирмы «Apple» Power Mac 8100 является также наиболее интересным компьютером, с точки зрения перспективности его использования в качестве высокопроизводительной рабочей станции, призванным составить конкуренцию компьютерам на базе Pentium. Сохранив почти все «изюминки» и преимущества Mac, став существенно быстрее, получив возможность запускать программы DOS и Windows, PowerMac вроде бы является наилучшим. Однако существует одно большое «но» — насколько быстро сумеют конкуренты выпустить нечто подобное и насколько быстро количество программного обеспечения в «родных» кодах PowerMac дойдет до необходимого уровня.

Передавая нам для тестирования PowerMac, фирма «Apple» предоставила наиболее быстрый на данный момент компьютер с тактовой частотой 110 МГц, что, естественно, не могло не сказаться на результатах. Укомплектован он был 16 Мб оперативной памяти, «шустрым» 1 Гб жестким диском, накопителем на компакт дисках, 17-дюймовым монитором, клавиатурой и однокнопочной «мышью». Ввиду поставки как компьютера, так и ПО от одной фирмы, проблем в процессе тестирования не возникло.

Остановимся более подробно на результатах тестирования. Честно говоря, они впечатляют. Быстрота, с которой выполнялись операции по обработке графики, просто не оставляет шансов другим компьютерам. Как на конвертирование RGB, так и на цветоделение по каналам CMYK PowerMac затратил менее одной секунды. Несколько медленнее выполнялось конвертирование в CMYK, оно заняло чуть более 6 секунд, что, по крайней мере, раза в 4 лучше, чем результат, показанный MAC Quadra, и в два раза — чем PC. Еще более впечатляющие результаты были получены при тесте на векторную графику. Трехкратное превосходство при сравнении с PC и Quadra говорит о том, что лучшее на сегодняшний день в PowerMac — это работа с графикой (что в общем-то ожидалось, но чтобы настолько быстро!).

На тестах же с базами данных и обработке текста были получены не очень впечатляющие результаты. Если с импортом базы PowerMac справился очень удачно (15 секунд), то сортировка по одному из полей выполнилась пусть и в три раза быстрее, чем на Quadra (полторы минуты), но семь секунд на PC не оставляют никаких сомнений, насколько «эффективно» выполнена реализация алгоритма сортировки в кодах PowerMac. Такими же невысокими явились результаты при обработке текста. Если Microsoft Word (в родных для PowerMac кодах) выполнил операцию поиска и замены менее чем за 50 секунд, то результат, показанный Mac Write Pro и затянувшийся аж на четыре с половиной минуты, вызвал недоумение. Очевидно, обработка текстовой информации (по большому счету, и базы данных текстового формата можно считать таковыми) является «ахиллесовой пятой» Mac-компьютеров.

Что касается компиляции примера, то результат в 47 секунд не назовешь блестящим. Это быстрее, чем то же на Quadra, но попрежнему медленнее, чем на PC.

Скорость выполнения операции сканирования компакт-диска каких-либо положительных или отрицательных эмоций не вызвала. Результат был получен средний, в районе тринадцати минут.

Кроме того, любопытства ради было проверено быстродействие эмуляции Windows. Результат, который мы получили, был где-то на уровне 486SX/33. Однако вопрос скорее не в быстродействии, а в самом факте эмуляции. Не совсем понятно, зачем понадобилось эмулировать реальный режим. Даже наиболее неприхотливым текстовым процессорам, электронным таблицам требуется, как минимум, 2 мегабайта памяти и 386 процессор, не говоря уже о таких могучих «агрегатах», как Word, Excel и другие. До тех пор пока PowerMac не сможет работать в расширенном режиме Windows и обеспечивать приемлемое быстродействие, говорить о достижении совместимости с Windows преждевременно.

Оценивая результаты, показанные Mac Quadra, можно сказать, что подавляющее преимущество над PC, продемонстрированное в обработке графики, сократилось (особенно это касается растровой графики), а неторопливость в обработке текстовой информации и файлов баз данных несколько снижают практическую ценность Quadra.

Для задач же, связанных с графикой (в особенности это касается цветокоррекции, цветоделения, наложения фильтров и верстки), подготовке и показу демонстрационных роликов (в особенности мультимедиа) Mac Quadra является тем инструментом, при помощи которого можно без проблем и головной боли получать неплохие результаты.

PowerMac же производит впечатление не до конца отшлифованного алмаза. Самая быстрая обработка графики, высокое качество отображения информации и ... неторопливость в обработке текста, пусть здесь PowerMac и быстрее чем Quadra, но все же медленнее, чем PC. По-видимому, заложенные в этом компьютере блестящие возможности требуют все-таки более качественного программного обеспечения. Система 7 безусловно хороша, но ее явно недостаточно для «самовыражения» PowerMac. Здесь нужно что-то другое, более близкое к открытым системам, более качественные алгоритмы обработки текстовой информации, которые смогут использовать всю мощь, заложенную в наиболее перспективный на сегодняшний день микропроцессор.

«Классика»—Gateway2000

Компьютер фирмы «Gateway2000», известной как сборщик неординарных как по быстродействию, так и по дизайну компьютеров, сразу производит впечатление на тех, кто привык к уже начавшим надоедать PC в типовых однообразных коробках mini-tower. Собранный в стильном корпусе desktop, укомплектованный красивым 15-дюймовым экраном с плоским монитором Vivitron, поставляемый с бездной полезных программ и всеобъемлющей документацией, компьютер демонстрирует квалифицированный подход к процессу сборки. Заглянув внутрь корпуса (кстати, придется приложить определенные физические усилия, чтобы «отодрать» крышку), можно увидеть, что собранная на заказной плате с набором чипов «Intel» с локальной шиной PCI система представляет обширные возможности для расширения. Имеющихся на плате пяти 16-разрядных слотов и трех слотов PCI с лихвой хватит для самых изощренных модификаций, так как контроллеры портов и накопителей находятся на основной плате, и лишь один слот PCI занят видеоадаптером STB Systems на чипе S3 с видео-BIOS Powergraph. Внешняя память состоит из жесткого диска Caviar фирмы «Western Digital», 3-дюймового дисковода фирмы «Epson» и накопителя на компакт-дисках фирмы NEC. В качестве интерфейса как для жесткого диска, так и для компакт-диска выбран IDE. Комплект дополняют расширенная клавиатура с несколько непривычным расположением клавиш управления курсором и двухкнопочная «мышь» фирмы Microsoft. И клавиатура, и «мышь» подсоединяются к компьютеру при помощи разъема типа PS/2. Гарантия в два года и цена комплекта в 3690 долларов, высокое качество сборки и множество дополнений, начиная от диагностических программ, коврика для «мыши» и кончая документацией, дающей ответ практически на все вопросы, которые могут возникнуть в процессе работы, являются ориентиром для тех фирм, которые хотят занять достойное место на рынке компьютеров.

Состоящий из вроде бы стандартных узлов, которые можно встретить и у компьютеров других производителей, P4D-66 с 16 Мб ОП, 520 Мб жестким диском и 2 Мб видеопамятью показал при тестировании очень хорошие результаты. За исключением двух тестов в X-Windows, которые не прошли в связи с отсутствием поддержки данного типа накопителей в Linux, при сравнении с другими PC, P4D-66 показал 14 наилучших результатов из 18 тестов, оказавшись в некоторых тестах вдвое быстрее, чем компьютеры «CompuLink» (тесты на компиляцию и на обработку векторного изображения). В остальных тестах преимущество было не столь подавляющим, и все же результаты, показанные компьютером Gateway, позволяют сказать, что при равной цене с CLR, похожих комплектующих (жесткий диск в комплекте CLR объективно быстрее накопителя, установленного в Gateway) P4D-66 является наиболее быстрым из PC-компьютеров, участвовавших в нашем тестировании. В суммарной же оценке всех платформ компьютер был слабее лишь в обработке растровой и векторной графики, где трудно выдержать конкуренцию с PowerMac. Очевидно, что специалистам фирмы удалось «разогнать» в общем-то не самые наилучшие компоненты до внушительных скоростей.

Оценивая компьютер Gateway в целом, можно сказать, что высокое качество сборки, внушительные результаты при тестировании, продуманный (но отнюдь не бесспорный) подход к подбору комплектующих, позволили бы рекомендовать P4D-66 в качестве оптимального, если бы не чуть непривычная клавиатура, снижающая удобство работы.

Необходимое послесловие

Результаты, полученные в процессе тестирования (таблица 1), дали ответ на многие из вопросов, интересующих тех, кто по

роду деятельности связан с компьютерной отраслью.

Утверждение о фантастическом быстродействии PowerMac в общем-то подтвердилось результатами тестов на обработку графики, а качество отображения информации на дисплее можно считать близким к идеальному. Результаты могли быть еще выше, если бы в компьютере были установлены различные акселераторы, дополнительная кэш-память, увеличено ОЗУ (сказанное справедливо и для компьютера Quadra). Несмотря на некоторое сокращение разрыва между Mac и PC в области обработки графики и настольных издательств (появление на PC возможности цветокоррекции в графических редакторах, более совершенные алгоритмы для цветоделения и наложения фильтров), все же говорить о реальной конкуренции в указанных областях пока преждевременно. Ожидаемая же весной поддержка расширенного режима Windows, появление новых, еще более быстродействующих микропроцессоров PowerPC приведут, по-видимому, к росту конкурентоспособности PowerMac и в других областях обработки информации.

Выпуск же фирмой IBM перспективной и давно ожидаемой OS/2 3.0 Warp, наконец-то свободной от тянущей назад 16-разрядности и позволяющей задействовать всю мощь Pentium для достижения максимального быстродействия, реальные шаги к более прогрессивной RISC-архитектуре, открытым системам, к объектному проектированию, позволяющему создавать аппаратно независимые программные продукты, приведут к еще большему обострению межплатформенной конкуренции, в выигрыше от которой окажется потребитель. ■

Результаты тестирования *Таблица 1*

Вид тестирования	CLR Tseng	CLR S3	CLR Trident	Quadra	Power Mac	Gateway P4D-66
Windows 3.1						
Конвертирование в RGB	1.97	3.43	2.34	3.54	0.58	2.34
Конвертирование в CMYK	24.38	26.91	26.19	25.72	6.22	24.72
Цветоделение по CMYK	2.47	6.50	4.59	1.94	0.82	1.85
Загрузка векторного изображения	25.90	30.30	29.56	7.69	2.84	9.90
Загрузка базы данных	21.00	22.88	22.37	1.46.28	15.85	15.21
Сортировка по алфавиту	5.28	6.85	5.81	3.25.06	1.20.81	5.15
Компиляция (библиотеки на жестком диске)	35.94	38.97	36.84	2.07.28	47.28	19.29
Компиляция (библиотеки на CD)	1.59.53	2.03.75	1.59.63	-	-	39.59
Поиск и замена по тексту	1.10.53	34.93	17.59	4.11.25 (3.35.94)	48.18 (4.27.47)	12.37
Сканирование CD	-	13.45.00	13.09.00	10.51.00	13.55.00	10.05.00
X-Windows 5.2						
Загрузка X-Windows	12.63	13.28	8.69	-	-	6.19
Загрузка GIF	19.22	21.60	15.25	-	-	-
Постановка фильтра Emboss	43.72	46.90	22.91	-	-	-
Расчет и построение фрактала Plasma	1.00.13	1.03.91	34.94	-	-	28.83
OS/2 3.0						
Замена шрифта	57.12	53.84	52.35	-	-	43.56
Сканирование CD	14.22.00	14.02.00	19.11.00	-	-	11.15.00
Построение трехмерного изображения	14.59.38	14.57.94	14.56.25	-	-	7.52.59
Загрузка OS/2	42.28	39.52	44.71	-	-	40.97

В таблице используется формат:
мм.сс.дд,
где мм — минуты,
сс — секунды,
дд — десятые и сотые доли секунды.

Отзывы фирм - участников тестирования

Фирма «Классика»

Независимая Тестовая Лаборатория и журнал «Монитор» проводят интересную работу по тестированию современной вычислительной техники. Большой набор разнообразных тестов позволяет составить достаточно полное представление о возможностях машин, прошедших тестирование.

Наша фирма представила на испытания типовую машину хорошо известной фирмы «Gateway 2000», которая успешно работает на рынке США, находясь на уровне ведущих производителей компьютеров по количеству продаж, сервису, качеству и надежности поставляемой техники.

Признавая важность работы Независимой Тестовой Лаборатории, хотелось бы отметить, что тестирование предполагает сравнение различного оборудования в одинаковых условиях. Но оборудование не должно быть очень разным. Учитывая широкий спектр интересов читателей журнала, можно согласиться с выбором для сравнения компьютеров, находящихся «достаточно далеко» друг от друга как по известности фирм-производителей, так и по областям целесообразного применения.

Нашу фирму такой подход не вполне устраивает по очень простой причине: мы поставляем технику коммерческим банкам, крупным государственным учреждениям, которые осознают, что в целом эксплуатация качественного оборудования оказывается более выгодной, даже если сама машина стоит на 30-40% дороже. Поэтому, объясняя своим заказчикам преимущества машин фирмы «Gateway 2000», мы сравниваем их с машинами таких известных фирм, как IBM, «Hewlett-Packard», «Compaq», «Dell». При этом опереться можно только на результаты, приводимые журналом PC World, а также на перепечатки их статей в других журналах. В результате же тестирования компьютера «Gateway 2000», проведенного журналом «Монитор», желаемых сопоставлений мы не получили.

Еще одно замечание по работе Тестовой Лаборатории. Выбранный набор тестов дает возможность проверить работу машины во всех режимах. Это нужно, если заранее неизвестно, как будет использоваться компьютер. Однако наши заказчики, как правило, хорошо представляют, для какой работы он им требуется. Для Тестовой Лаборатории, наверное, не составит большого труда выяснить, какие операции и в каком процентном отношении представлены в том или ином пакете, и получить индекс каждой машины для конкретных приложений.

Начальник отдела сетевых и информационных технологий
Попов Е.Н.

НПО «Компьюлинк»

Тестирование графических станций Независимой Тестовой Лабораторией обрадовало нас своей жесткой программой. CLR-компьютер оказался достаточно быстрым, и наши надежны оправдались. Подтвердилась также полная совместимость узлов этой конфигурации при работе под DOS, Windows, OS/2. Возникшие проблемы с установкой Linux помогли выявить несовместимость в выбранной конфигурации кэш-контроллера и мультикарты GoldStar, которая в другой конфигурации не проявляется.

В целом считаем участие в таких тестированиях неотъемлемой частью своей технической политики и предпочитаем массовой рекламе обеспечение нашего потенциального потребителя достоверной и объективной информацией.

В дальнейшем мы предполагаем подвергнуть жесткому тестированию CLR-компьютер с PCI и Pentium-90.

Технический директор
Трубецкой Никита

RUI Apple Computer

Тестовая Лаборатория и журнал «Монитор» проводят важную работу по сравнению вычислительной техники различных производителей.

Однако хотелось бы обратить внимание на ряд недостатков:

- для тестирования Power Macintosh были использованы Excel и Word, которые «по определению» работают медленнее на Макинтоше, так как они были изначально предназначены для работы в среде Windows и лишь затем перенесены на MacOS;
- у нас было недостаточно времени для поиска «родных» программ для PowerMac, в результате чего пришлось тестировать программы для 680x0 процессоров, что, естественно, отразилось на результатах.

Безусловно, фирма RUI Apple Computer IMC заинтересована в продолжении сотрудничества, но нам хотелось бы заранее быть ознакомленными с программой тестирования, чтобы успеть подготовить соответствующее программное обеспечение (или аналогичное ему по возможностям).

Менеджер отдела новых продуктов
Олег Котик

Комментарий НТЛ

Прозвучавшая в отзывах на тестирование критика нас обрадовала. Если бы тестирование было заурядным, оно не вызвало бы столько вопросов.

Еще раз повторим, что мы не ставили своей целью последовательный перебор программ и компьютеров, определение «наилучшего в открытии файлов», «наименее зависающего», «чемпиона по соотношению «цена/производительность/доступность/гарантия» и прочего.

Мы специально выбрали компьютеры столь разного класса, поскольку интерес представляло именно то, насколько велика «дельта» в полученных результатах. Что касается программного обеспечения, то оно было выбрано стандартным для каждой платформы, а с программой тестирования все участники были ознакомлены предварительно.

НТЛ считает своей задачей не присвоение почетных званий «чемпиона», «лидера в своем классе» или «лучшей покупки», а максимально ответственную и жесткую проверку функционирования компьютеров с последующей полной публикацией результатов.

Подписку на журнал «Монитор» на 1995 год можно оформить в фирме «Информ»: 113447 Москва, а/я 3, тел. *(095) 129-68-29.*

Для тех, кто имеет возможность забирать журналы в редакции, можно оформить подписку в АОЗТ «Монитор». Для этого надо приехать к нам, заполнить купон, заплатив по наличному расчету. Цена годовой подписки 12$ в рублях по курсу ММВБ на день оплаты. Вы будете получать бесплатно электронную версию каждого выпуска на Вашей дискете.

Адрес редакции:
Москва, ул. Большая Якиманка, 38А, 4 этаж.
Телефон: *238-99-11, 238-99-33, 238-31-89*. Факс: *238-99-11*.
E-mail: post@mntr.msk.su.

Михаил Сычев
принимает посетителей
по вопросам подписки
в редакции
по четвергам
с 11 до 17 часов.

Храните ваши байтики...

Сегодня мы вновь продолжаем разговор о тех технических решениях, которые используются фирмой СКТБ Компьютерных Сетей в своих разработках [1], [2]. Интенсивные исследовательские работы в области создания современных моделей персональных компьютеров в фирме не прекращаются ни на день. И рост популярности торговой марки CND в значительной мере обусловлен тем, что за пять лет работы на рынке вычислительной техники специалисты СКТБ КС сумели добиться значительных успехов в обеспечении высокого качества выпускаемых изделий, положив в основу своих моделей достижения многомесячного труда исследовательского отдела.

Полученные результаты позволили довольно своеобразно отметить юбилей фирмы, установив на компьютеры марки CND пятилетнюю гарантию, что фактически соответствует сроку морального старения изделия. И немаловажную роль при определении столь длительного срока гарантийной службы сыграл выбор надежных и высокопроизводительных устройств поддержки массовой памяти, именуемых в просторечии винчестерами.

Нет нужды рассказывать читателям «Монитора» о различиях между интерфейсами SCSI и AT. А вот какие конкретно модели используются в компьютерах CND, имеет смысл рассмотреть более подробно. Но вначале немного о ходе исследований.

Винчестеры

Производителей винчестеров известно немало. Нет причин не доверять репутации столь известных фирм, как IBM или «Seagate». Но российская действительность вносит свои коррективы в расстановку сил на отечественном рынке. Вот, например, с чем столкнулись специалисты СКТБ КС при комплектовании компьютеров высокопроизводительными винчестерами с интерфейсом SCSI, имеющими объем 1 Гб и среднее время поиска 9,5 мс.

Попытки воспользоваться изделиями столь хорошо известного производителя, как IBM, особого успеха не имели. Проблема оказалась в чрезмерном «интеллекте» интегрированного контроллера винчестера. Пользователи оставались недовольны тем, что винчестер живет своей «внутренней жизнью», выполняя какие-то операции по собственной инициативе, что, согласитесь, вызывает неприятные мысли. Заказчик, как известно, всегда прав, и поэтому пришлось искать другие варианты.

Винчестеры фирм «Quantum» и «Micropolis» также показали себя далеко не с лучшей стороны. Не желая оскорбить производителей, отметим, что в поставляемых в Россию партиях SCSI-винчестеров процент брака составляет от 10 до 20 процентов, что согласитесь, не способствует повышению надежности и снижению себестоимости готовых изделий.

Никаких проблем не возникло только с винчестерами производства фирмы «Conner». Низкое энергопотребление, малый уровень шума, довольно высокая производительность и минимум настроечных переключателей, а также понятная и доступная документация сразу же выделили эти изделия среди конкурентов, а практически полное отсутствие заводского брака оказалось решающим фактором, который в конце концов и склонил чашу весов в пользу Conner. Похожая ситуация сложилась и с IDE-винчестерами. Так, компьютеры, которые оборудовались винчестерами производства фирмы «Western Digital», через полгода оказались в гарантийной мастерской. А надежные 2,5" накопители фирмы IBM емкостью 170 и 82 Мб, как оказалось, предполагают, что при расширении системы (то есть при подключении второго винчестера, выбор будет сделан только в пользу IBM. В противном случае доступа к slave-винчестеру вы просто не получите!

Никаких нареканий не вызвали винчестеры фирмы «Maxtor», но все же, с точки зрения критерия «эффективность/стоимость», винчестеры фирмы «Conner» и в этом классе накопителей оказались вне конкуренции. Летнее падение цен на рынке комплектующих позволило СКТБ КС отказаться от морально устаревших накопителей емкостью менее 340 Мб и в то же время не допустить повышения цен на компьютеры. В таблице 1 приведены основные технические характеристики 3,5" винчестеров Conner FilePro, которые покрывают весь спектр потребностей в устройствах массовой памяти.

Но собственно винчестер — это только половина дела. Для того чтобы обеспечить доступ к данным, нужен еще и соответствующий контроллер. Поскольку стоимость контроллеров, как минимум, на порядок меньше стоимости винчестеров, имеет смысл подбирать интерфейсные платы, которые обеспечивают наилучшие результаты при работе с выбранными типами винчестеров. Давайте взглянем, чего достигли в СКТБ Компьютерных сетей.

Мультиплаты с IDE-контроллером

Что ни говори, интерфейс IDE по-прежнему остается самым популярным. В результате предложение мультиплат с интегрированным IDE-контроллером на рынке исключительно велико. Но выбрать действительно надежную плату оказалось не так-то просто. Рассмотрим вначале мультиплаты для шины ISA.

На сегодняшний день при всем многообразии производителей этих относительно несложных устройств все платы построены на одном из двух интерфейсных наборов: Goldstar Prime 2C и Winbond.

Платы на основе Goldstar достаточно популярны. К сожалению, ряд конструктивных недостатков проявляется в самый неожиданный момент.

Статья подготовлена с использованием материалов, предоставленных фирмой СКТБ Компьютерных сетей.

Публикуется на коммерческой основе.

Фирма «Goldstar» при разработке набора Prime 2C, видимо, просто забыла учесть необходимость управления шинными формирователями. В результате в платах на основе этого набора нет шинного усилителя для восьми старших разрядов IDE, которые коммутируются прямо на шину ISA. Нет также буферизации и двунаправленного порта принтера. А это приводит к росту нагрузки на шину ISA, что отрицательно сказывается на надежности материнской платы.

Значительно интереснее мультиплаты, построенные на основе интерфейсного набора фирмы «Winbond». Наряду с полной буферизацией сигналов, в зависимости от установленных микросхем (таблица 2) платы Winbond могут поддерживать и высокоскоростные порты, и накопители высокой плотности.

Кроме того, если с некоторыми материнскими платами контроллеры Prime 2C по ряду причин «не уживаются», то Winbond работает без нареканий со всеми моделями — от PC/AT до Pentium. Именно поэтому модели CND Junior комплектуются контроллерами на основе интерфейсного набора Winbond.

Если же вы приобретаете компьютеры серии Gepard (на базе 80486 с шиной VESA Local bus), то для обеспечения максимальной производительности дисковой подсистемы целесообразно воспользоваться более современными контроллерами, использующими преимущества, которые предоставляет локальная шина.

Строго говоря, до последнего времени на этом секторе рынка особого выбора не было — практически на всех платах использовался интерфейсный набор Promise 2, который на сегодняшний день можно считать устаревшим. Дело в том, что стремительный прогресс в технологии производства жестких магнитных дисков привел к значительному росту объемов накопителей и оживлению рынка IDE-дисков и IDE-контроллеров, который начал было сдавать позиции под натиском со стороны SCSI. Но вот незадача — практически все IDE-контроллеры предусматривают подключение не более двух жестких дисков, в то время как возможности SCSI..., впрочем, о SCSI поговорим позже.

Проблема была замечена, и через некоторое время на рынке появились материнские платы, BIOS которых поддерживала управление четырьмя IDE-устройствами. Но ведь поддержки со стороны BIOS недостаточно для работы с четырьмя винчестерами! Нужен и соответствующий контроллер. Именно поэтому в последние модели компьютеров Gepard устанавливается контроллер MIO-V2 на базе интерфейсного набора VIA 82C461.

Конечно, четыре винчестера вам, возможно, устанавливать и не потребуется. Но IDE, как мы уже упоминали, получает второе дыхание, что приводит к появлению новых периферийных устройств, подключаемых к дисковому контроллеру. За примерами далеко ходить не надо — это высокопроизводительные стримеры и приводы для чтения CD-ROM.

АППАРАТУРА

Полная буферизация мультиплаты позволяет
- снизить нагрузку на шинные формирователи материнской платы;
- защитить микросхемы дорогостоящей материнской платы от внешних помех (в том числе и высоковольтных);
- снизить нагрузку на блок питания.

Но вот незадача! Для того чтобы использовать возможности, предоставляемые VLB-контроллерами, вам потребуются специальные драйверы. В противном случае никакого реального увеличения производительности вы не получите. Как ни странно, большинство контроллеров поставляется без соответствующего программного обеспечения, что, прямо скажем, вызывает у пользователя разочарование. И это еще одна из причин, по которой в компьютеры серии Gepard устанавливаются контроллеры MIO-V2, — в комплект поставки входит набор драйверов для MS-DOS, Windows, Windows NT, UNIX SCO и Novell Netware, что позволяет пользователю эффективно работать с любой операционной системой.

Контроллеры SCSI

Несмотря на то, что с появлением современных IDE-контроллеров и быстродействующих IDE-накопителей интерфейс SCSI сегодня не воспринимается как единственный путь расширения возможностей системы, хоронить его пока рано. Причин тому несколько.

Во-первых, SCSI еще далек от «последнего издыхания» и продолжает довольно динамично развиваться. Пример тому — появление новой версии SCSI-2, отличающейся большей производительностью по сравнению с первой реализацией.

Во-вторых, устройства с интерфейсом SCSI могут подключаться не только к PC совместимым компьютерам, но и к широкому спектру устройств других производителей — от Macintosh («Apple») до Iris («Silicon Graphics»). В результате выбор периферийных устройств с интерфейсом SCSI оказывается настолько необъятным, что альтернативу ему в ближайшие годы будет подыскать весьма непросто.

В-третьих, SCSI выступает как расширитель ограниченных возможностей шины персонального компьютера, позволяя подключать к одному контроллеру до восьми периферийных устройств.

И в-четвертых, производительность SCSI-интерфейса выше, чем IDE, что оказывается особенно привлекательным при построении сетевых серверов.

На сегодняшний день фактическим стандартом дискового SCSI-контроллера является Adaptec 1542. Но, как известно, любая идея порождает большое число подражаний, причем в ряде случаев копии оказываются удачнее оригиналов.

Поэтому, не желая завышать цену конечного изделия, специалисты СКТБ КС предприняли исследования плат SCSI-контроллеров, совместимых с Adaptec 1542 и поддерживающих интерфейс ASPI на уровне ПЗУ BIOS. В ходе исследований оказалось, что наилучшее соотношение «эффективность/стоимость» обеспечивают платы SCSI 742 производства фирмы FIC (Тайвань), той самой, которая поставляет материнские платы для компьютеров CND [1]. Кратко перечислим основные особенности семейства FIC SCSI 742:
- 32-разрядный интерфейс шины;
- 16-разрядный микропроцессор, управляющий работой контроллера;

- поддержка режима прямого доступа к памяти со скоростью передачи до 33 Мб/с;
- скорость передачи данных до 10 Мб/с;
- поддержка системы команд SCSI-2;
- управление парой НГМД 5,25/3,5";
- многозадачный режим работы (до 255 задач);
- встроенное ПЗУ BIOS для однозадачной среды MS-DOS.

Контроллеры SCSI 742 выпускаются для всех типов шин: ISA, VLB, EISA. При этом сохраняется полная идентичность с точки зрения программиста. А для удобства эксплуатации контроллеры имеют как внутренние, так и внешние разъемы.

Но необходимо отметить, что ПЗУ контроллеров совместимы с ранними версиями Adaptec 1542 и содержат код загрузки операционной системы, а также поддержку вектора 13h, что позволяет работать с DOS без использования дополнительных драйверов и резидентных программ. Без каких-либо проблем контроллер работает под управлением ОС Linux (напомним, что поддержка интерфейса SCSI реализована в ядре системы). Но так же, как и с Adaptec 1542, возникают проблемы с Windows NT и Windows for Workgroups.

Последние версии драйверов Adaptec, которые ориентированы на работу с данными программными продуктами, с семейством FIC SCSI 742, к сожалению, не совместимы. Но разработчики программного обеспечения уже создали драйверы и для этих плат. Поскольку СКТБ КС поддерживает самые тесные связи с изготовителем контроллеров, новые версии драйверов оказываются доступны практически немедленно после выпуска, что позволяет организовать полноценную поддержку потребителя.

Контроллеры для шины PCI

Было бы странно, если бы в СКТБ КС, приступив к выпуску компьютеров серии Pelican на базе процессора Pentium, не использовали возможности новой шины PCI. И хотя выбор PCI-устройств на рынке, особенно в России, пока не слишком велик, решение было найдено. В результате в состав компьютеров включен современный SCSI-2 контроллер с пиковой пропускной способностью 110 Мб/с, поддерживающий программные интерфейсы ASPI и SDMS CAM. В состав программного обеспечения, поставляемого с контроллером, входят драйверы для MS-DOS (на базе ASPI), Windows 3.0 и выше, Windows NT 3.1, OS/2 2.0, Netware 386 4.0 и 3.11 и SCO UNIX 3.2 и 4.x.

Основные параметры винчестеров серии FilePro *Таблица 1*

Модель	CFA340A	CFA420A	CFA540A
Тип интерфейса	IDE	IDE	IDE
Емкость (после форматирования)	343 Мб	426,8 Мб	541 Мб
Среднее время поиска:			
с дорожки на дорожку	3 мс	3 мс	3 мс
среднее (чтение/запись)	13 мс	14 мс	12 мс
максимальное	25 мс	25 мс	26 мс
Среднее время задержки	7,5 мс	8,3 мс	6,7 мс
Скорость вращения	4011 об/мин	3600 об/мин	4500 об/мин
Скорость передачи данных	7,5 Мб/с	7,5 Мб/с	10 Мб/с
Надежность:			
среднее время наработки на отказ	300000 часов	250000 часов	300000 часов
срок службы	5 лет	5 лет	5 лет
Надежность хранения данных	не более одной ошибки на 10+14 битов		

Как оценить платы Winbond? *Таблица 2*

Микросхема	Характеристики
W83757AF	2*IDE + 2*FDD(360 кб - 1,44 Мб) + 2S (16450/9600 Бод) + 1P + 1G
W83787F	2*IDE + 2*FDD(360 кб - 1,44 Мб) + 2S (16550/112000 Бод) + 1P + 1G
W83777F	2*IDE + 2*FDD(360 кб - 2,88 Мб) + 2S (16450/112000 Бод) + 1P + 1G

Выбор накопителей и контроллеров, как вы видите, дело далеко не праздное, если вы организуете серийный выпуск компьютеров, которые должны решать самые разнообразные задачи. Судя по всему, за надежность работы системы массовой памяти в компьютерах CND можно быть спокойным. Но это еще не все. Наряду с качественными комплектующими, в обеспечении надежности вычислительных систем важную роль играет комплексная система качества выпускаемой продукции. О том, как этот процесс организован в СКТБ Компьютерных Сетей, вы узнаете в следующих номерах. ■

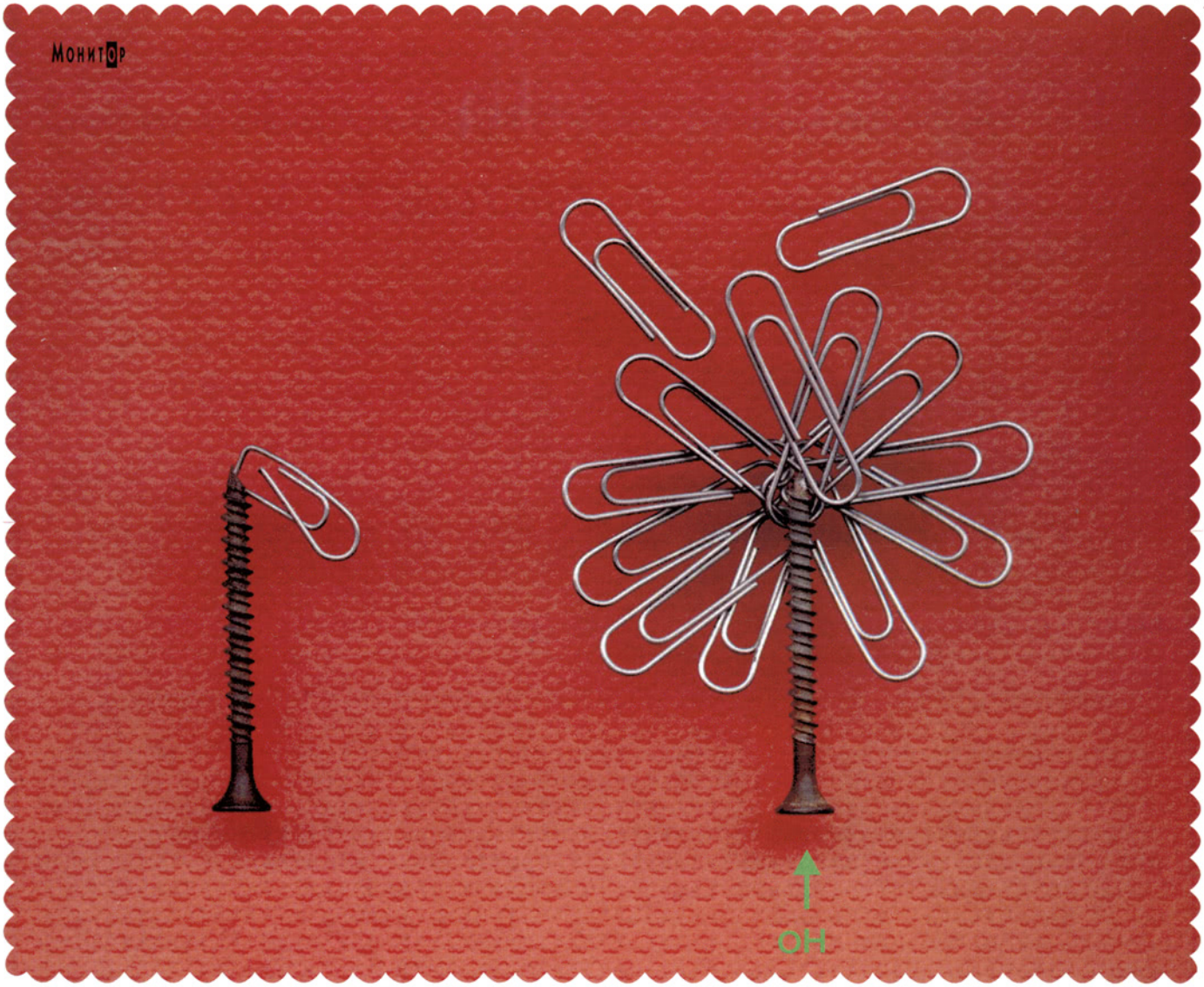
Литература

1. **Кто в тереме живет?// Монитор.- 1994.- № 6.**
2. **Все краски радуги.... Монитор.- 1994.- № 7.**

КОНТАКТЫ

Магазин-салон СКТБ по продаже вычислительной и оргтехники

г. Москва,
Севастопольский проспект, 43б.
Телефоны: (095) 127-9011, 127-6850.
Факс: (095) 127-7990.
E-mail: cnd@network.msk.su



дал

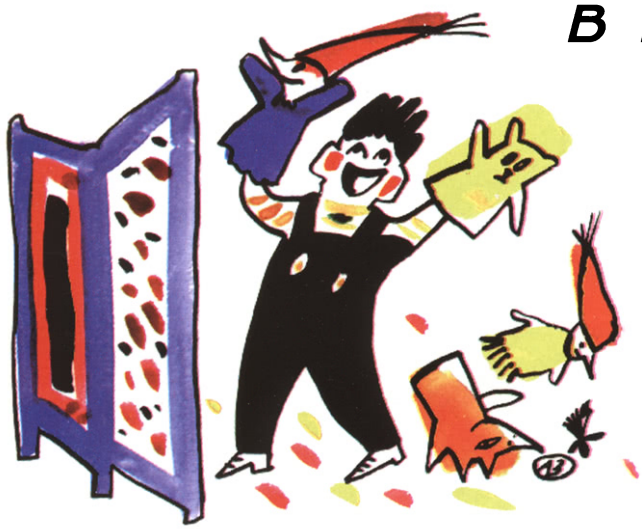
рекламу

в журнале

«МОНИТОР»

Для тех, кто понял - телефоны отдела рекламы журнала «Монитор»: 903-46-26, 238-99-11.

Базовые средства мультимедиа в Microsoft Windows



В.Дьяконов

Установки в Windows для работы с мультмедиа, работа с базовым программным обеспечением.

Как настроить Windows под мультимедиа

Если вы намерены работать с мультимедиа в среде графической оболочки Windows, то вам потребуется ПК класса не ниже 386SX с ОЗУ 4 Мб и более и тактовой частотой не менее 10 МГц. Необходима также плата видеоадаптера класса SVGA с разрешением 640×480 пикселей при 256 цветах (такая плата должна иметь видеоОЗУ хотя бы на 512 килобайтов). Именно таковы параметры платы и дисплея, используемые в большинстве видеоприложений мультимедиа под Windows. Windows должна работать в расширенном режиме.

Один из важных моментов работы с мультимедиа в среде Windows заключается в установке нужных драйверов дисплея и звуковой платы. Однако надо отметить, что не всегда такая установка проходит гладко, — особенно если нужных драйверов нет на инсталляционных дискетках системы Windows. К примеру, автору пришлось потратить немало времени на поиск драйвера к видеоадаптеру RealTek для режима SVGA с 256 цветами. Следует внимательно отнестись и к установке драйверов звуковой платы — случается, что они конфликтуют с ранее загруженными драйверами в среде MS-DOS.

Для установки драйверов надо использовать пиктограмму Drivers в окне Control Panel. Это приведет к появлению диалогового окна Drivers со списком уже установленных драйверов (рис. 1). Этот список и определяет, какие конкретно устройства можно использовать в мультимедиа.

Можно удалить отмеченный драйвер, указав курсором «мышки» клавишу Remove. Можно, напротив, добавить в список драйверов новый, указав клавишу Add. При этом появится окно Add со списком добавляемых драйверов (справа от окна Drivers на рис. 1).

Обратите особое внимание на первую позицию списка драйверов в окне Add — Unlisted Updated Drivers (драйвера нет в списке или обновить драйвер). Если сделать эту позицию активной (клавиша Ok), то можно вызвать соответствующее диалоговое окно, а из него — окно Browse для поиска файлов по всем директориям и дискам (рис. 2). Это необходимо, если искомый драйвер отсутствует в главной директории Windows.

Обнаруженный драйвер загружается и появляется в окне Drivers. Перед этим система запрашивает, нужен повторный старт Windows или нет. Для инсталляции драйвера рестарт обязателен. Но если вы инсталлируете несколько драйверов подряд, то рестарт можно дать после установки последнего драйвера. Это несколько ускорит процесс инсталляции драйверов.

Рекомендуется иметь набор драйверов с определенным запасом. Так, полезно иметь драйверы дисплея VGA (640×480 точек при 16 цветах), VGA Monohrome (640×480 точек при 2 цветах) и SVGA (640×480 точек при 256 цветах). Первый драйвер можно применять при повседневной работе, а последний — при просмотре видеоклипов. Монохромный драйвер полезен, если вы го-

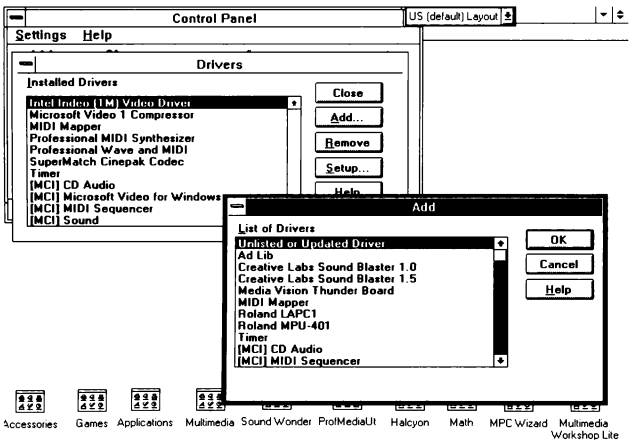


Рис. 1. Обзор драйверов при их инсталляции

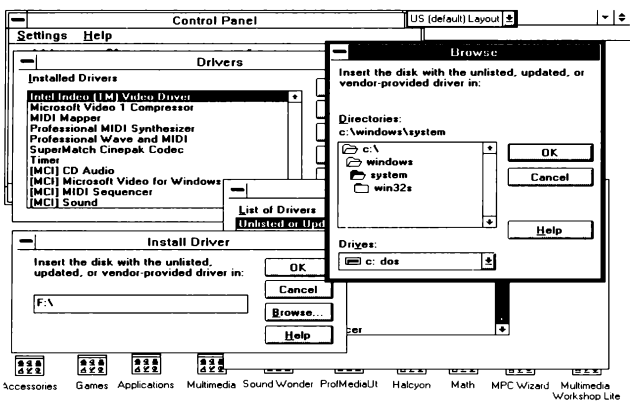


Рис. 2. Поиск нужного драйвера

товите отчеты или пишете статьи и книги — пока, увы, в них вставляются почти исключительно копии экранов с черно-белым изображением. Перестройка драйверов, когда они установлены и имеют статус Current, занимает несколько секунд, вместе с рестартом оболочки Windows. При этом нет нужды вновь разыскивать и использовать гибкие дискетки с драйверами. Бывают случаи конфликтов драйверов с некоторыми мультимедиа-приложениями, и это тоже основание для того, чтобы иметь несколько драйверов на всякий пожарный случай.

Озвучивание оболочки

Если вы установили драйверы звука, то можете неожиданно обнаружить, что загрузка и выход из оболочки Windows, а также остановки и вывод сообщений об ошибках сопровождаются мелодичными звуками. Словом, происходит озвучивание оболочки — первый признак перехода к работе с мультимедиа.

Концепция озвучивания оболочки базируется на связи происходящих при работе с оболочкой событий с некоторым набором звуковых файлов с расширением WAV. Эту связь устанавливает диалоговое окно Sound, пиктограмма которого находится в окне Drivers. Если активизировать пиктограмму, то можно увидеть это окно (рис. 3).

В левой части окна Sound находится список событий, которые можно выделять цветовой меткой. В данном случае выделено событие Windows Start (старт оболочки Windows). В правой части окна расположен перечень звуковых файлов с расширением WAV, которые хранятся в главной директории Windows по умолчанию. Можно отметить файл, который должен озвучивать систему, если произойдет выбранное событие.

Как вы, вероятно, догадались, можно дополнить список звуковых файлов своими файлами и связать их с теми или иными событиями. К примеру, вы можете связать старт системы с файлом, в котором записана фраза «Вас приветствует система Windows», а выход из системы — с файлом, дающим сообщение «Система Windows прощается с Вами». Остальное зависит уже только от вашей фантазии. Как создавать свои звуковые файлы, будет описано немного далее.

Выбор средств мультимедиа

С позиций мультимедиа, Windows 3.1 является платформой, которая базируется на специальном интерфейсе для связи между средствами мультимедиа — MCI (Media Control Interface). Он предполагает использование следующих средств, заданных своими именами:

cdaudio	—	проигрыватель звуковых компакт-дисков,
dat	—	цифровой магнитофон,
digitalvideo	—	средства цифрового видео,
movie	—	проигрыватель мультмедиа фильмов,
overlay	—	адаптер Video Overlay,
scanner	—	оптический сканер,
sequencer	—	MIDI-секвенсер (синтезатор),
ver	—	видеомагнитофон,
videodisc	—	видеодиск (видеопластинка),
waveaudio	—	аудиоадаптер.

Обозначение [MCI] входит в название соответствующего драйвера. Например, драйвер для работы со звуком имеет название [MCI] Sound. Наличие таких драйверов для используемых устройств обязательно. Установка драйверов описывалась ранее.

В самой Windows средства мультмедиа представлены всего четырьмя программами, расположенными в окне Accessories (рис. 4). Это программы Recorder, Media Player, Sound Record и

Object Packager. Тем не менее это небольшое число программ позволяет выполнять основные функции мультимедиа — от проигрывания звуковых файлов и аудиодисков до создания своих простых видеороликов со звуковым сопровождением. Создается впечатление, что эти средства имеют скорее учебный, чем профессиональный характер и призваны заинтересовать пользователя в серьезном применении мультимедиа с соответствующими, более серьезными средствами.

С помощью указанных программ вы можете работать с видеомагнитофоном (Record), проигрывать файлы различных типов (Media Player), записывать и исполнять звуковые файлы (Sound Record) и объединять объекты для их совместного применения (Object Packager).

Проигрыватель файлов Media Player

При всей своей внешней простоте проигрыватель мультимедиа файлов — многофункциональное «устройство». Мы в первый раз взяли это слово в кавычки, поскольку в действительности речь идет о программе, а не о реальном устройстве. Но степень имитации реальных устройств в мультимедиа столь высока, что далее мы будем именовать их без кавычек.

Итак, если вы сделаете активной пиктограмму Media Player в окне приложений Accessories, то увидите появление небольшого изображения проигрывателя мультимедиа файлов (рис. 5). Оно поразительно напоминает изображение автомобильного проигрывателя — магнитофонного или компакт-дисков. В частности, внизу видны кнопки с обычными для таких устройств функциями пуска, перемотки, остановки и так далее.

От автомобильного проигрывателя мультимедиа плеер отличается строкой в верхней части с позициями основного меню. На рис. 5 она активна и выводит список устройств, функции которых может выполнять мультимедиа плеер. Еще раз отметим, что они определяются теми драйверами, которые вы инсталлировали в системе. К примеру, если вы хотите, чтобы плеер превратился в проигрыватель звуковых лазерных дисков, то укажите тип устройства CD Audio в позиции выбора устройства Device главного меню. Примечательно, что наш плеер может проигрывать даже видеофайлы с динамической графикой!

Но вернемся пока к звуковым файлам довольно распространенного формата MID (стандарт MIDI). Попробуем исполнить эти файлы с помощью мультимедиа плеера. Для этого надо выбрать устройство MIDI Sequencer в позиции De-

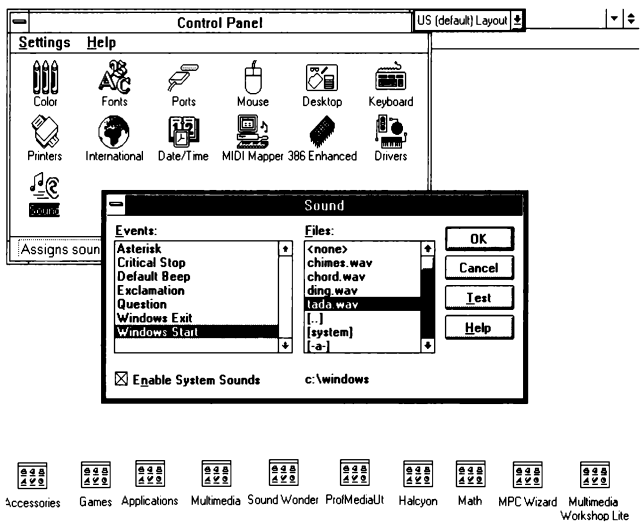


Рис. 3. Окно связи событий оболочки Windows со звуковыми файлами

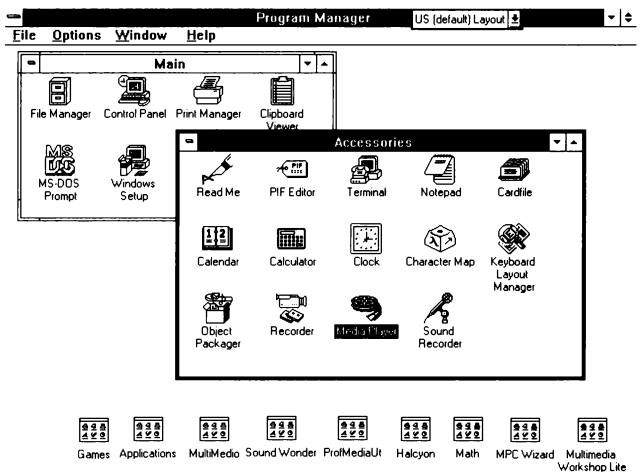


Рис. 4. Окно Accessories с основными приложениями мультимедиа

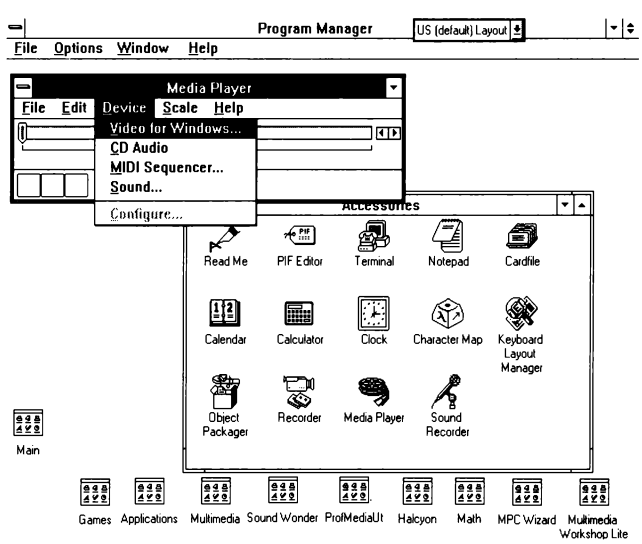


Рис. 5. Мультимедиа плеер с открытым окном выбора устройств

vice главного меню плеера. Далее для выбора требуемого файла нужно активизировать подпозицию Open (открыть) в позиции File главного меню. Откроется обычное для всех приложений Windows диалоговое окно для выбора файлов (рис. 6).

Как только вы загрузите файл, станет заметно «оживление» плеера — очертания некоторых кнопок сменятся их четкими жирными изображениями, подсказывая, что плеер готов к работе (рис. 7). Теперь стоит нажать клавишу пуска (это первая клавиша слева), как можно увидеть перемещение ползунка по линейке плеера — он указывает относительное время проигрывания. Если с инсталляцией драйверов у вас полный порядок и ПК оснащен звуковой платой, то вы услышите звуки, которые были записаны в выбранный файл.

С проигрыванием файлов MIDI связаны определенные проблемы. Дело в том, что эти файлы вовсе не несут информации о звуковых колебаниях как таковых. Такую информацию несут лишь файлы с расширением WAV (от слова wave — волна). Файлы типа MIDI управляют работой встроенного в плату звука музыкального синтезатора. Поэтому они несут закодированную информацию о том, какие инструменты должны звучать, в какое время, с какой интенсивностью и по каким нотам. В принципе, такая информация занимает гораздо меньший объем, чем подвергнутая цифровому кодированию обычная речь или музыка.

Беда, однако, в том, что различные производители звуковых плат, музыкальных клипов и записей плохо договорились друг с другом о том, каким конкретно номерам должны соответствовать различные инструменты в синтезаторах. В результате, при неправильной настройке ноты для скрипки могут исполняться роялем и так далее. В окне Control Panel имеется (если установлен соответствующий драйвер) пиктограмма с именем MIDI Mapper, которая (при необходимости) позволяет провести правильную настройку порядка расположения инструментов. Для этого она выводит диалоговое окно с набором имен инструментов. Разумеется, для настройки с помощью этой программы необходимы определенные профессиональные знания в области музыки.

Столь же просто можно проиграть небольшой готовый видеоролик. Для этого установите устройство Video for Windows и найдите файл с расширением AVI. Его нет в стандартной поставке Windows, но множество таких файлов поставляется с видеодрайверами или имеется на компакт-дисках для вашего CD-ROM. Пустив плеер, вы увидите под ним окошко с изображением (рис. 8).

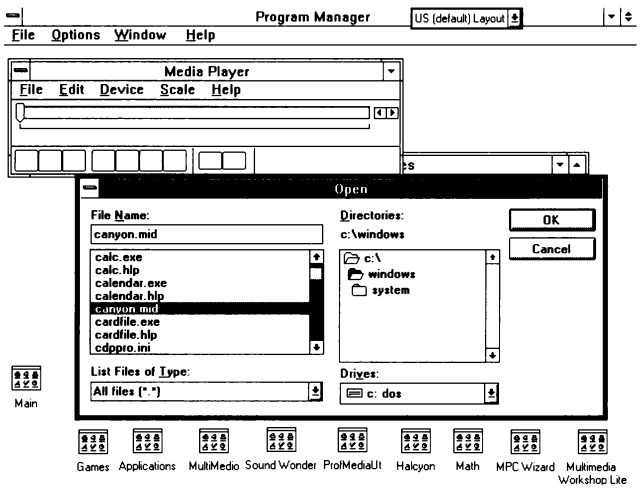


Рис. 6. Окно выбора файлов мультимедиа плеера

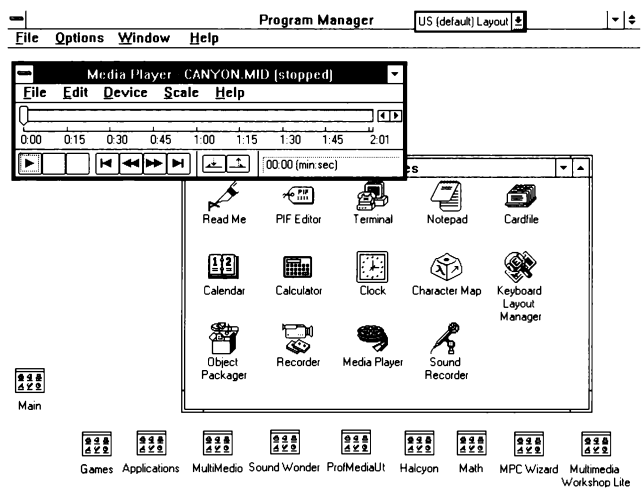


Рис. 7. Мультимедиа плеер в режиме проигрывания звуковых файлов

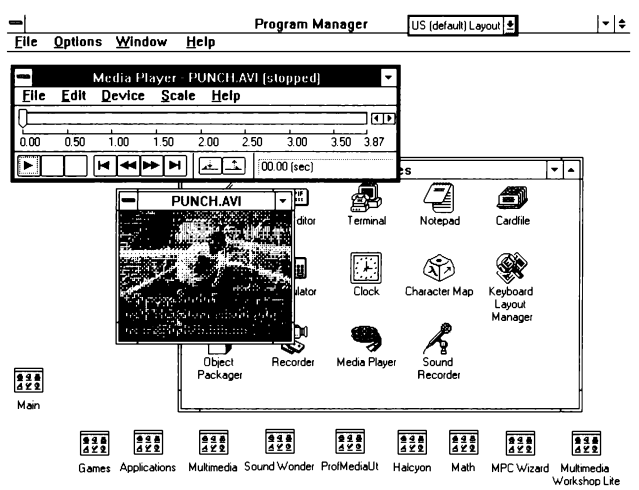


Рис. 8. Мультимедиа плеер в режиме просмотра изображения

Представленный на рис. 8 замороженный черно-белый стоп-кадр небольшого видеоклипа едва ли потрясет чье-либо воображение — разве что лишь принципиальной возможностью такой демонстрации. На нем с трудом можно разглядеть человека, стоящего у автомобиля. Однако если изображение цветное, то четкость изображения гораздо выше. В этом режиме можно наблюдать динамические изображения — например, при пуске будет видно перемещение человека. В целом, однако, базовые средства мультимедиа в Windows не предназначены для создания высококачественных изображений. Куда совершеннее средства использования звука!

АППАРАТУРА

Графическая оболочка Microsoft Windows реализует почти все возможности мультимедиа, используя всего четыре базовые программы-утилиты.

Запись звука и изображения

При внимательном просмотре панели мультимедиа плеера можно обнаружить кнопку записи. С ее помощью можно записать небольшой (обычно до 60 секунд) звуковой фрагмент. Однако сам плеер в действительности лишь воспроизводит аудио и видеофайлы. При записи он фактически использует другую, вполне самостоятельную программу, пиктограмма которой имеет надпись Sound Recorder — в буквальном переводе «записыватель» звуков. Эта программа имитирует действие обычного магнитофона, создает и проигрывает аудиофайлы.

Рис. 9 иллюстрирует работу с Sound Recorder в режиме поиска звукового файла с расширением WAV.

Действие программы зависит от того, какая «кнопка» активна. Если активна кнопка воспроизведения, то программа исполняет загруженный файл и его звучание можно услышать. Программа создает изображение осциллографа, в котором можно увидеть осциллограмму звукового сигнала (рис. 10). Это позволяет вести контроль за его уровнем и оценивать степень искажений.

Если задать запись (активна кнопка записи), то можно произвести запись звука в новый файл или в уже существующий. Источником сигнала при записи является микшер, так что запись можно провести с микрофона, линии, синтезаторов музыки и голосов и CD-проигрывателя. Программа Sound Recorder позволяет также редактировать звуковые файлы, например встроить в файл запись новых звуков или скопировать файл с другим уровнем звука и временной задержки. В последнем случае можно получить эффект эха.

Если ПК оснащен видеоплатой для работы с видеомагнитофоном, то можно произвести и запись телевизионных изображений. Это выполняет программа Recorder, пиктограмма которой имеет изображение видеокамеры. При пуске этой программы появляется диалоговое окно управления (рис. 11).

Работа с этой программой достаточно очевидна, но следует отметить, что видеоплатами для работы с видеомагнитофонами пока оснащено весьма небольшое число ПК.

Связывание объектов и программа Object Packager

Одной из важнейших черт Windows является возможность представления всех устройств, которые имитируются программными средствами, в виде работающих объектов. Они представляются пиктограммами соответствующих программ.

В позиции Edit главного меню каждого приложения есть команда Copy, заносящая объект в буфер Clipboard, а также команда Paste, вызывающая объект из буфера и вставляющая пиктограмму выбранного объекта в заданное положением курсора место. Если на такую пиктограмму установить курсор «мышки» и дважды щелкнуть левой клавишей, то будет запущен подчиненный объект. Таким образом можно, например, вставить в текст редактора Write ссылку на какой-нибудь рисунок, подготовленный в графическом редакторе PaintBrush.

Разработчики программного обеспечения всеми силами стараются предусмотреть такую возможность в своих прикладных программах. Есть она и в большинстве приложений, поставляемых с системой Windows, в том числе и в приложениях, связанных с мультимедиа. Поэтому можно в текст редактора Write вставлять пиктограммы, запускающие звуковые плееры, создающие необходимые звуки. Они должны быть представлены соответствующими файлами, загруженными в мультимедиа средства, например в проигрыватель Sound Recorder.

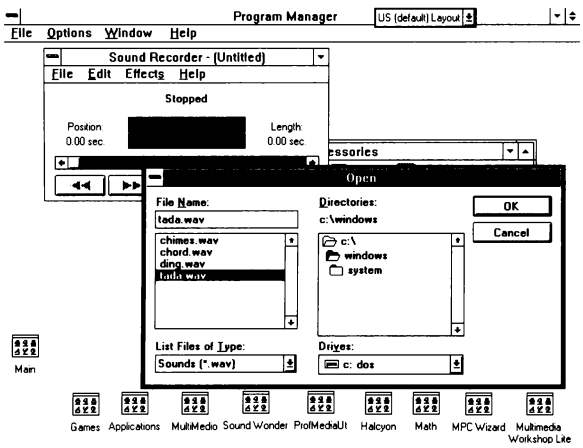


Рис. 9. Работа с программой Sound Recorder

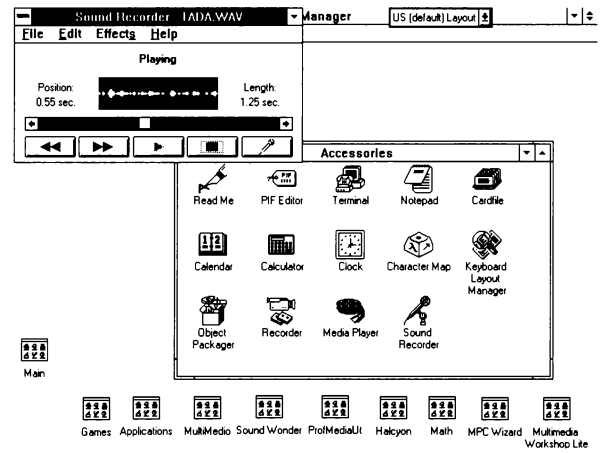


Рис. 10. Воспроизведение звукового файла программой Sound Recorder

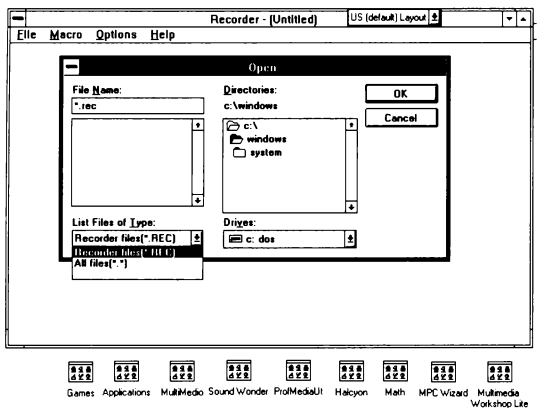


Рис. 11. Диалоговое окно программы Recorder

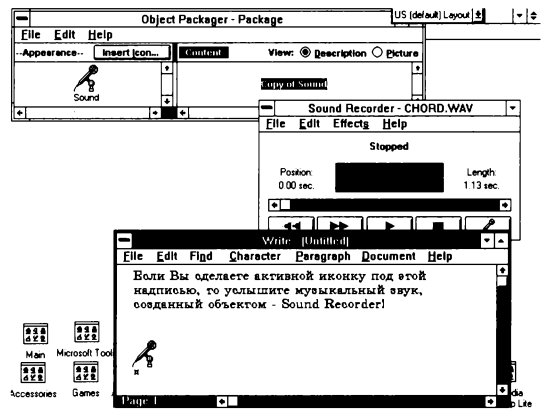


Рис. 12. Пример связывания текста с озвучивающим его объектом

Рис. 12 иллюстрирует технику связывания текстов с проигрываем звуковых файлов. Вначале надо подготовить такой файл с помощью проигрывателя Sound Recorder или загрузить в него подходящий из имеющихся файлов. После этого в позиции Edit главного меню надо сделать активной позицию Copy. В результате этого в буфер будет скопирован объект — проигрыватель звуковых файлов с загруженным в него файлом.

Теперь надо запустить приложение — текстовый редактор Write и ввести в него текст, предшествующий вызову объекта. Затем, установив курсор в нужное место, надо сделать активной команду Paste в позиции Edit главного меню этого приложения. В результате, что показано на рис. 12, в тексте появится пиктограмма, активизация которой приведет к исполнению выбранного звукового файла.

В состав Windows входит специальная программа Object Packager, позволяющая связать воедино ряд объектов. Так, на рис. 12 показан результат связывания двух объектов — текстового редактора Write с введенным в него текстом и проигрывателя звуковых файлов с введенным в него файлом. Для связывания Object Packager с приложением Sound Recorder надо также использовать команду Paste в позиции Edit главного меню программы Object Packager. Пример такого связывания приведен на рис.12.

Динамическое связывание объектов и его глубинный смысл не сразу постигается пользователями, привыкшими к идеологии MS-DOS приложений. Поэтому надо поупражняться в создании динамически связанных приложений. Полезно дополнительно прочитать об этом в книгах [1] и [2]. ■

Литература

1. **Стефан Фойц. Windows 3.1/ Пер. с нем.- Киев: BHV, 1993.**
2. **Михаэль Кирмайер. Мультимедиа/Пер. с нем.- С.-Пб.: BHV, 1993.**

Струйный принтер—экзотика или рабочий инструмент?

В. Водолазкий

Нет нужды доказывать читателям «Монитора», что принтер — это не роскошь, а средство повышения производительности труда. И вполне очевидно, что хочется получить результат (выходной документ) получше, а затратить труда и средств поменьше. Но чудес, как известно, не бывает. И в результате на столах программистов и пользователей господствуют два основных класса устройств печати — матричные и лазерные принтеры.

Наибольшее распространение получили в России «демократичные» матричные принтеры. Но, увы, невысокое качество печати, низкая скорость вывода данных и шум при работе являются неизбежным злом, с которым приходится мириться, приобретая недорогую модель. А средне- и особенно высокоскоростные матричные принтеры конкурируют по цене со своими лазерными собратьями.

Кстати, о лазерных принтерах. Качество печати, конечно же, не в пример лучше. И работают они почти бесшумно. Но за все приходится платить, в буквальном смысле этого слова.

А можно ли объединить и качество печати, и низкий уровень шума, и производительность, и невысокие цены? Оказывается, можно. И сегодня речь пойдет о таких принтерах, а именно — о струйных.

– Стоп, стоп, стоп, — скажет искушенный читатель. — Но ведь струйные принтеры, как известно, исключительно тихоходные. А если еще учесть, что кириллицы в них нет и печатать придется в графическом режиме...

Спору нет, читатель, как всегда, прав. Поэтому сделаем еще одно уточнение. Речь пойдет о струйных принтерах фирмы «Olivetti». На сегодняшний день на российский рынок продвигаются три модели струйных принтеров, которые заслуживают вашего внимания.

Но прежде чем говорить об особенностях каждой из моделей, давайте разберемся, что их объединяет.

Во-первых, все они обеспечивают печать с разрешением от 300×300 точек/дюйм до 600×300 точек/дюйм у модели JP450, что, как вы понимаете, не уступает качеству печати принтера Hewlett-Packard Laser Jet IVL. Кроме разрешения, принтеры серии JP (jet printer) совместимы с лазерными принтерами HP и по языку управления, поскольку эмулируют язык PCL 3+.

Во-вторых, принтеры содержат шрифты с кириллицей для всех начертаний, поддерживаемых этими моделями. Не считая различных модификаций (наклонный, полужирный, с подчеркиванием и так далее), вы получаете возможность работать с пятью гарнитурами: Courier, Letter Gothic, Book Face Times, Times Nordic, Linea и Line Draw. При этом поддерживаются как кодовая страница 866, так и кодировка MS Windows. И это далеко не мелочь! Средняя стоимость процесса «русификации» струйного принтера по Москве составляет сегодня 40 долларов. Так что, если вы не хотите ограничиваться возможностями Лексикона при работе в DOS, поинтересуйтесь у продавца, поддерживается ли кириллица в выбранной вами модели. И если нет, прибавьте в мыслях указанную сумму к стоимости изделия.

Что касается Windows, то тут, на первый взгляд, проблем значительно меньше. Шрифты TTF одинаково (или почти одинаково) отображаются на любом устройстве вывода. Но за это приходится платить скоростью печати. Так вот, модели серии JP позволяют загрузить до 25 TTF-шрифтов во внутреннюю память принтера, а кроме того, аппаратно выполняют преобразования текста при различных форматах листа (portrait/landscape), что и позволяет добиться эффективной работы под управлением Windows даже на относительно слабых машинах.

Да, третье преимущество струйных принтеров фирмы «Olivetti» — это скорость печати. Она, конечно, варьируется для разных моделей и составляет от полутора страниц в минуту для JP 50 до пяти у модели JP 450. А ведь «страницы в минуту» — это уже терминология из области лазерных печатающих устройств!

И еще один важный момент. Бичом любого принтера в России является доступность расходных материалов к нему, но только не для «Olivetti» и его генерального дистрибьютера в России «ART trading, Co.». И в особенности это относится к струйным принтерам. Ресурс сменного картриджа (чернильницы) составляет в среднем 400000 символов при печати в режиме LQ (Letter Quality). Но много это или мало? Давайте попробуем оценить количество страниц. И поскольку мы сравниваем струйные принтеры с лазерными, то и будем использовать соответствующие методики оценки.

Как правило, за эталонный лист принимается заполненность области печати — 25% (поскольку оформление при подготовке выходных документов играет далеко не последнюю роль). Каждый символ, в свою очередь, «залит» краской в среднем на четверть. И полагая, что на листе формата A4 мы можем разместить 64 строки по 80 символов, мы получим, что пять процентов от объема листа составят всего 1280 символов.

Имейте в виду, что ресурс кассет с тонером тоже рассчитывается, исходя из этой методики! В конце концов, она придумана не мной, и вот уже несколько лет кочует по всей технической документации к лазерным принтерам. Так вот, исходя из этих оценок, мы получаем, что ресурс картриджа составляет ни много ни мало — 312,5 страниц. Как показывает практика, обычной кассеты матричного принтера хватает примерно на 200 листов, что,

Статья подготовлена с использованием материалов, предоставленных фирмой «Olivetti». Печатается на коммерческой основе.

по крайней мере, в полтора раза меньше. И можно было бы размахивать этими цифрами как флагом, если бы мы не решили сравнивать струйные принтеры Olivetti с лазерными. Давайте оценим полную стоимость печати листа (без стоимости бумаги) для принтеров серии JP и популярного в России лазерного принтера HP LJ IVL.

В таблице 1 приведены ваши затраты на эксплуатацию принтера, исходя из ежемесячного расхода 300 листов. Это означает, что в течение месяца вы ежедневно печатаете 10 листов (то есть примерно 12 кб) или с учетом выходных — по 12,5 листов в рабочие дни.

Как видите, совершенно неожиданно выясняется, что струйный принтер экономичнее! Конечно, через полтора года сумма общих затрат сравняется, но только в том случае, если вы изо дня в день будете «испекать» по десять страниц.

Теперь позвольте сказать пару слов об удобстве работы. В последние годы на российском рынке появилось большое количество матричных принтеров нового поколения, которые позволяют использовать самую разнообразную бумагу и имеют несколько механизмов подачи. Фирма «Olivetti» также не осталась в стороне от этого процесса. И в результате струйные принтеры JP обеспечивают протяжку рулонной бумаги, автоматическую загрузку листовой бумаги (автоподача до 70 листов) и, кроме того, загрузку листа из передней части принтера. При этом листовая бумага печатается, конечно же, без перегиба, что особенно важно для различного рода финансовых документов. Нет нужды говорить, что такой подход позволяет наиболее эффективно использовать поверхность рабочего стола, которой, как всегда, оказывается недостаточно. Но, похоже, об этом в «Olivetti» думают давно и всерьез, потому что серия JP выпускается в компактных корпусах с плавными обводами без того обилия выступающих частей и рукояток, на которые постоянно натыкаешься при работе с матричным принтером.

И последняя особенность, присущая струйным принтерам Olivetti. Вы не задумывались, почему цветной струйный принтер стоит почти в два раза дороже черно-белого? А в «Olivetti» задумались. И в результате любая (кроме JP50) модель принтера превращается из черно-белой в цветную путем замены баллончика с краской (65$) и установки в соответствующее положение одного из переключателей на самом принтере. Ну и, конечно же, необходим соответствующий драйвер, но он поставляется в комплекте с принтером.

Теперь первое знакомство с серией JP можно считать состоявшимся, и имеет смысл кратко коснуться особенностей каждой из моделей.

Первая модель — JP 50 — это портативный принтер весом чуть больше килограмма, предназначенный для работы «в полевых условиях». Скорость печати — 100 символов в секунду — соответствует средним характеристикам матричных принтеров, но при этом никакого шума и качество печати лазерного принтера. Под управлением Windows JP-50 обеспечивает производительность до 2 страниц в минуту. Несмотря на свои габариты (130×300×60 мм), принтер позволяет заправлять бумагу не только по одному листку, но и в лоток. Ресурс печатающей головки в этой модели ниже — всего 200000 символов, но это плата за компактность изделия. Несмотря на это, JP-50 оказывается отличным принтером для notebook и не потребует для переноски дополнительного чемодана. Как и notebook, принтер может работать от встроенных аккумуляторов. Средняя планируемая нагрузка на принтер — 5 листов в сутки.

Модель JP360 представляет собой центральную модель семейства. Содержит 128 кб оперативной памяти, которая используется не только как входной буфер, но и для хранения TTF-шрифтов. Обеспечивает вывод данных на бумагу, пленки, конверты. Средняя производительность под управлением Windows составляет 2,5-3 страницы в минуту, а при символьной печати — 180 знаков в секунду в режиме Draft и 120 знаков в секунду в режиме LQ. Номинальная нагрузка — до 350 страниц в день и не более 1400 страниц в месяц. Но это понятно. Принтер — все-таки не типография! При этом потребляемая от электрической сети переменного тока мощность не превышает 18 Вт. И тут еще одна статья экономии!

И, наконец, модель JP450. Это самый быстрый струйный принтер Olivetti, который обеспечивает скорость до 400 знаков в секунду в режиме Draft! В Windows, конечно, производительность поскромнее — до 5 страниц в минуту в режиме Draft, до 4 — в режиме NLQ и «всего» около 3 в режиме LQ. При этом нагрузка на принтер, как и в прошлой модели, не должна превышать 350 страниц в день. Не стоит забывать, что эти 350 страниц, как и в JP360, могут быть цветными, что делает последние две модели незаменимыми при подготовке полноцветных изданий. Интересно, а во сколько вам обойдется цветной лазерный принтер? И будет ли он потреблять от сети переменного тока 25 Вт, как JP450? А вот еще одна интересная особенность JP450. Это режим «экономии чернил», который автоматически включается при заливке графических областей. Мелочь, конечно, но позволяет сэкономить до 10% чернил при работе с графикой.

И последний момент, который может оказать решающее значение для приобретения струйных принтеров JP. Кто, где, почем... Ответить на этот вопрос очень легко. Генеральным дистрибьютором «Olivetti» в России является фирма ART trading Co., которая обеспечит поставку любых моделей. Компания расположена в центре Москвы, но как дистрибьютор заинтересована в формировании региональной дилерской сети. А почем... Так ведь ответ на этот вопрос следует из пожеланий, изложенных в самом начале статьи: качество лазерного принтера при цене матричного! ■

ART trading Co.
Москва, Новая площадь 8/10.
Телефон: *(095)921-9686.*
Коммерческий отдел:
(095)921-2622

Оценка годовых затрат на печать данных *Таблица 1*

Тип принтера	JP 360	Hewlett-Packard Laser Jet IVL
Стоимость принтера	351$	700$
Ресурс головки/кассеты (листов)	312	7500
Стоимость головки/кассеты	8$	100$
Месячный расход на эксплуатацию	7,7$	4$
Годовые расходы	92,4$	48$
Общие расходы за год	443,4$	748$

АНОНС

«Монитор»—версия 3.95

А. Анненков
Вычисление числительных, или о пользе рекурсии
С. Ерёмин
Еще раз о TSR...
А. Костылев
О скоростном заполнении областей экрана в трехмерной графике
В. Медведев
Еще раз о палитре адаптера VGA
А.Мурадян
Визуальное программирование
Часть 2.
Microsoft Visual C++ 1.5—AppWizard, ClassWizard, AppStudio, MFC (Microsoft Foundation Classes)
А. Козленков
Универсальная система ведения программных проектов
IBM Workframe/2 2.1
М. Рожков
«Чистое» кодирование на C++
Часть 1.
Распространенные ошибки
С. Королев
Разработка нестандартных объектов в Super Vision 3.0
М. Трофимов
Жизнь на Макинтоше: по страницам новой книги
Часть 2
С. Абрамова, А. Замятин
Использование специализированной библиотеки B-Tree Filer
Ю. Крючков
Novell DOS: что новенького?
В. Плотников
Описание метода разработки драйвера внешнего устройства для Windows
Н.Косарев, Ю.Крючков, В.Хорт
Открытое письмо Президенту всемирно известной «Microsoft Corporation» Биллу Гейтсу
С. Короткий
Введение в распознавание образов.
Часть 3. Выделение углов
А. Боресков, Е. Шикин
3D Studio: работа с текстурой
В. Дьяконов, С. Бирюков
Derive в России
А. Дубина, С. Сынгур
Впервые ли за день включен ПК?
В.Дьяконов
Царство мультимедиа в Microsoft Windows

МИНИ-АТС 1x6

Мини-АТС GT-HS106
идеальна для дома
или небольшого офиса

цена - от $81 до $376 !!!

тел. (095)325-0088,322-0879;
факс (095)325-2293
ЛАТЕК

Отличное качество звучания
Великолепная эргономика
Оригинальный дизайн
Необычные функции

Изменение голоса, удаленное прослушивание

ТЕЛЕФОНЫ

ЛАЗЕРНЫЕ ТЕХНОЛОГИИ в РЕКЛАМЕ

ВЫВЕСКИ и ТАБЛИЧКИ
из металла и пластика
световые и несветовые
ЛАЗЕРНАЯ ГРАВИРОВКА
по любым материалам
на сувенирной продукции

Московский Центр Лазерной Технологии

(095) 361-9695 (095) 362-5617

ARUS MOSCOW

HEWLETT PACKARD
Authorized Wholesaler

НАДЕЖНОСТЬ, ПРОВЕРЕННАЯ ВРЕМЕНЕМ

Компьютеры, серверы, сканеры, лазерные и струйные принтеры, сетевое оборудование фирмы *HEWLETT-PACKARD*

Наши региональные дилеры:

Москва:	**"ARUS-Тех. Центр"**	(095) 316-7627	(095) 316-8328
	"K&M"	(095) 323-9366	(095) 323-9172
	"E+E"	(095) 916-1449	(095) 917-3379
	"EuroNet"	(095) 281-0178	(095) 281-4141
	"DiViSy"	(095) 460-4723	(095) 460-0212
	"Синэус"	(095) 318-0722	(095) 318-6324
	"Delta Group"	(095) 241-6968	(095) 241-6968
	"Смарт"	(095) 556-4379	(095) 556-4262
Ангарск:	**"Форт-Диалог"**	(39518) 5-7653	(39518) 6-3216
Казань:	**"Абак"**	(8432) 322-023	(8432) 323-061
Наб. Челны:	**"Форт-Диалог"**	(8439) 56-1467	(8439) 56-8129
Новосибирск:	**"Велл"**	(3832) 328-857	(3832) 391-527
Одесса:	**"ARUS-Odessa"**	(0482) 668-090	(0482) 657-055
Самара:	**"Резольвента"**	(8462) 222-996	(8462) 222-996
Тольятти:	**"Пролог +"**	(8469) 346-730	(8469) 341-183
Уфа:	**"Арти-Системс"**	(3472) 35-6280	(3472) 33-0535
Юж.-Сахал.:	**"Сахин"**	(42422) 31-478	(42422) 31-397

Продолжаем формирование дилерской сети

316-7627, 316-8328, 110-5830, 230-6808. Fax: 119-6841

Гарантия HEWLETT-PACKARD до 3-х лет

Сканеры фирмы UMAX

Д. Мокин

Многие сталкивались со сканерами, но даже высокообразованные читатели «Монитора» не всегда хорошо представляют разницу в возможностях и назначении разных моделей. Мы хотим рассказать о новой производственной программе фирмы UMAX. Название фирмы в нашей стране, возможно, больше знакомо тем, кто работал на Macintosh или серьезно занимался издательской деятельностью, хотя UMAX выпускает множество моделей для самых различных приложений и разных платформ.

Сначала — экскурс в теорию. Чем различаются сканеры? Во-первых, конструкцией: они бывают планшетными и барабанными, и если с первыми читатели «Монитора» наверняка знакомы, то со вторыми сталкивались далеко не все. Барабанные сканеры сложнее в обслуживании, медленнее, точнее и дороже. Несколько лет назад они считались единственно пригодными для серьезной работы с цветом, однако сегодня это уже не так. Во-вторых, сканеры бывают цветными или черно-белыми, цветные же бывают однопроходными или трехпроходными, да и способность восприятия цветов у них разная. И в-третьих, все сканеры имеют разную разрешающую способность. По поводу разрешающей способности существует много спекуляций, связанных с тем, что кроме истинной (оптической, физической) разрешающей способности, определяемой оптикой, размерностью CCD-линейки, точностью привода существует еще интерполяционная разрешающая способность (более высокая), показывающая лишь, сколько придуманных точек может поместить программное обеспечение между физическими. Зачастую в документации к дешевым сканерам не акцентируется, о каком разрешении идет речь. Конечно, и хорошая интерполяция может дать полезные результаты, особенно, если речь идет о сканировании чертежей. Однако в приложениях, связанных с обработкой фотографий, важно лишь оптическое разрешение.

Общее для всех сканеров UMAX то, что они планшетные.

Начнем со сканера **UMAX PowerLook**, самой популярной модели фирмы. Это аппарат, позволяющий во многих случаях удовлетворить нужды настольного издательства, работающего с цветом, и обойтись без помощи барабанного аппарата. Он имеет оптическое разрешение 1200×600 dpi (более высокого у планшетных сканеров дешевле $20 000 и не бывает) и внутреннее представление цветовой информации с размерностью 30 битов; цветные оригиналы считываются за один проход. Кроме того, на него стандартно устанавливается адаптер для сканирования слайдов. Особый интерес представляет система преобразования цвета: дело в том, что стандартные форматы хранения изображений позволяют записывать только 24 бита цветовой информации, и самые распространенные программы, такие как Adobe Photoshop, тоже работают с 24 битами. При первом сканировании (его можно сделать с низким разрешением) сканер линейно преобразует внутреннее 30-битовое представление во внешнее 24-битовое (естественно, часть информации при этом теряется — 6 битов просто отбрасываются). Теперь пользователь имеет сведения об изображении, о его оптической плотности и может создать функцию преобразования для «чистового» сканирования, которая позволит добиться наилучшей передачи деталей в самых критичных местах слайда и исправить ошибки экспозиции при фотографировании. Затем эта функция загружается в сканер, и при повторном сканировании он использует ее для преобразования данных. Программа **MagicScan,** поставляемая вместе со сканерами UMAX, позволяет упростить и частично автоматизировать процесс создания функций преобразования, контроля передачи цветов и многое другое. Другая полезная функция, реализованная в

Статья печатается на коммерческой основе.

сканере и в программе, — decreening, размывание растров. Часто приходится работать не с фотографическими, а с полиграфическими оригиналами, после сканирования которых изображение оказывается покрытым сеткой. Этот эффект возникает в результате взаимодействия двух регулярных объектов: полиграфического растра и механики и оптики сканера. Удалить такую сетку с изображения довольно сложно, поэтому UMAX имеет специальный режим, в котором она подавляется уже на аппаратном уровне; сканирование происходит медленнее, но зато изображение получается без дефектов. Еще одно свойство UMAX PowerLook — уникально высокий для планшетного сканера динамический диапазон оптических плотностей 3.0D. Существует много барабанных сканеров, у которых эта величина не превышает 2.8D, а у большинства планшетных она около 2.5D. Наличие перечисленных характеристик позволяет относить UMAX PowerLook к классу издательских сканеров. Действительно, он дает возможность обрабатывать чаще всего встречающийся в серьезных работах объект — слайд с весьма приличным качеством в отношении цветопередачи, а его разрешающая способность гарантирует, что при полиграфическом воспроизведении слайд выдержит увеличение в 3-4 раза. При формате слайда 6×7 см и формате издания A4 этого почти всегда достаточно, а такие форматы чаще всего используются в издательской работе.

Следующая модель фирмы UMAX — **Vista S8**. Этот сканер появился в конце 1994 года и призван удовлетворить потребности тех, кто нуждается в высококлассном сканировании, но не может пока позволить себе PowerLook. Vista S8 — цветной однопроходный 24-битовый сканер с оптическим разрешением 800×400 dpi. Он также может оснащаться адаптером для слайдов, но может поставляться и без него; при желании вы приобретаете адаптер позже. Можно сказать, что этот аппарат находится на границе между дорогими офисными и дешевыми издательскими моделями, при этом по цене он ближе к первым. Особо интересны заложенные в него и в поставляемую с ним версию программы MagicScan возможности программно-аппаратной интерполяции: они обеспечивают интерполяционное разрешение до 6400 dpi! Такие высокие величины возможны благодаря тому, что привод и оптика сканеров Vista работают с очень высокой стабильностью, и поэтому получаемая от них информация обладает высокой линейностью и практически не нуждается в дополнительной коррекции.

Близкий родственник этого сканера — **UMAX Vista S6**. Он отличается несколько меньшим разрешением — 600×300 dpi и меньшей ценой. По цене и по оптическому разрешению он близок к популярному Hewlett-Packard IIcx, но по деловым качествам заметно превосходит его. Во-первых, он работает быстрее, особенно в режимах максимального разрешения, в которых некоторая «сонность» HP IIcx проявляется сильнее всего. Во-вторых, и это более важно, хотя его привод и оптика выдают примерно столько же информации, как и у HP IIcx, но, как и в случае с Vista S8, эта информация более надежна и линейна и является намного лучшей базой для интерполяции. Как следствие — более высокая точность при большом разрешении. Кроме того, таких возможностей, какие предоставляет программа MagicScan, обеспечение, поставляемое с HP IIcx пока не дает. Vista S6 также может оснащаться адаптером для слайдов.

Последняя из выпускаемых сегодня моделей — **UMAX Vista T-630**. Этот аппарат имеет общую с Vista S6 механику и оптику, разрешение 600×300 dpi, 24 бита, но только одну CCD-линейку, и, как следствие, цветные изображения он считывает за три прохода. Такой сканер, благодаря невысокой цене, удобен для тех, кому цветные изображения приходится обрабатывать редко, и с потерей скорости можно мириться, но хотелось бы при этом иметь хорошее качество. Так же как и остальные модели UMAX, он может оснащаться дополнительными устройствами: адаптером слайдов и устройством автоматической подачи документов (ADF). Причем для этой модели ADF весьма актуально: благодаря невысокой цене и большой скорости обработки монохромных изображений оно (как, впрочем, и Vista S6) прекрасно подходит для систем «безбумажного офиса».

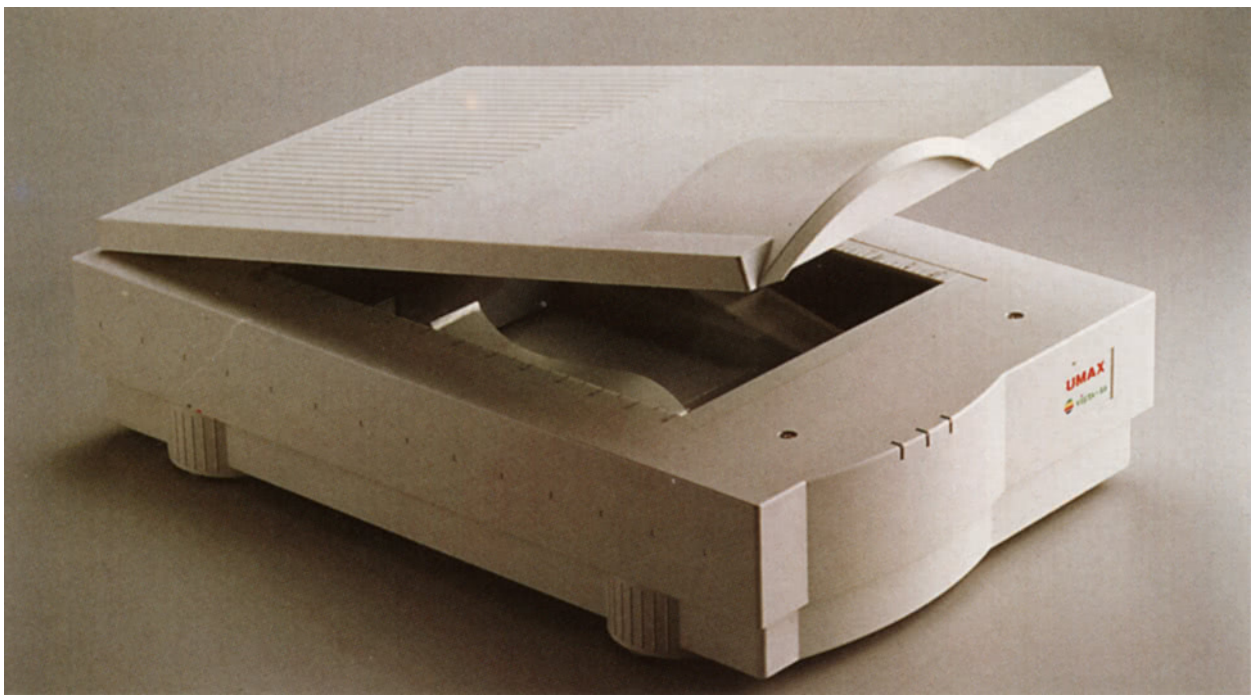
И еще две модели, пока не имеющие названия, будут представлены фирмой UMAX в этом году. Во-первых, дешевый аппарат, замыкающий модельный ряд снизу: офисный монохромный сканер с разрешением 300 dpi, 256 градаций серого, обрабатывающий при этих параметрах 10 страниц A4 в минуту. И во-вторых, сканер с предварительным названием UMAX Gemini D16, в котором объектив автоматически меняется в зависимости от формата обрабатываемого изображения. Такой сканер позволит слайд размером 6×7 см увеличивать при полиграфическом воспроизведении в 6-8 раз, при этом оставаясь в 3 раза дешевле барабанного сканера с аналогичными возможностями и намного удобнее в обращении. Мировая премьера этого аппарата намечена на май 1995 года. Официальным представителем UMAX в России является фирма YAM International, в которой можно получить более подробные консультации и приобрести эти сканеры. ■

Телефоны 334-2344, 334-2801.

UMAX

ЛУЧШИЕ В МИРЕ ЦВЕТНЫЕ ПЛАНШЕТНЫЕ СКАНЕРЫ

ИХ ОФИЦИАЛЬНЫЙ ДИСТРИБУТОР В СНГ



Коллекция новейших моделей:

- Vista-T630
  Экономичный цветной сканер для офисов. Скорость - до 13 страниц А4 в минуту.
- Vista-S6 и S8
  Идеальные сканеры для настольных издательских систем. Считывание в один проход.
- PowerLook
  Профессиональный сканер для издателей. 30 бит. Диапазон оптических плотностей - 3.0. Включает приставку для сканирования слайдов.

Признанные наиболее удобными для пользователей пакеты сканирования,калибровки цвета и обработки изображений:

- MagicScan
- MagicMatch
- PhotoShop LE/FE

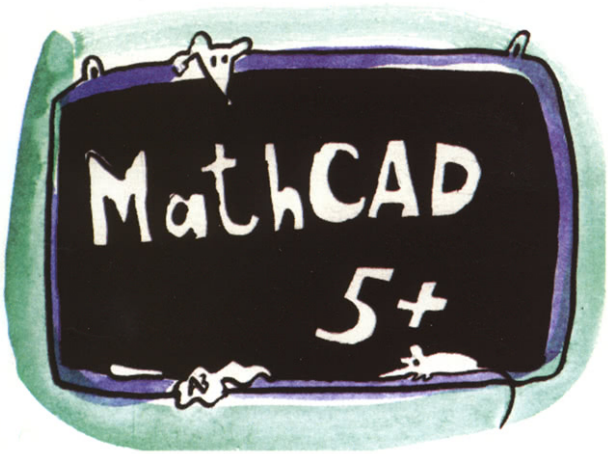
По желанию: устройство автоматической подачи оригиналов,приставка для слайдов.

Цены - начиная с $895. Поставка со склада в Москве.

YAM International:
Ул. Профсоюзная,57,Москва.
Тел.: 334-23-44. Факс: 334-28-01.

Приглашаем дилеров!

MathCAD — пять с плюсом!



В.Дьяконов

Новейшая версия интегрированной математической системы MathCAD PLUS 5.0 вполне заслуживает своего названия. Ей можно выставить оценку «пять с плюсом» за превосходный интерфейс, высокую скорость работы и многочисленные новые возможности.

Лучшее — друг хорошего?

Многие разработчики математических систем долгое время руководствовались житейской мудростью «лучшее — враг хорошего». Фирме «MathSoft, Inc.» эта «мудрость» явно не присуща. За какие-то три года эта фирма не только затолкала свой популярный продукт Mathcad [1], [2] в оболочку-бестселлер Microsoft Windows, но и ухитрилась выдать «на гора» целых четыре его версии под Windows — от вызвавшего восторги MathCAD 3.0 [3] до новейшей версии MathCAD PLUS 5.0, появившейся в середине 1994 г. Подобной плодовитостью пока не могут похвалиться ее конкуренты.

Да, фирма «MathSoft, Inc.» в очередной раз показала, что она вовсе не намерена уступать свое обжитое место многочисленным конкурентам, наводняющим рынок программного обеспечения математическими системами [2]. Отослав читателя к врезке по новой системе MathCAD 5.0, рассмотрим, что же нового привнесло слово PLUS в сверхновейшую популярную математическую систему для всех — MathCAD PLUS 5.0.

Выставляем пять с плюсом!

Многие математические системы страдали общим пороком — чем выше порядковый номер версии, тем медленнее становилась их работа. Особенно на младших собратьях нового поколения 32-разрядных персональных компьютеров (ПК) — машинах класса 386. Было очевидно, что разработчики систем делали ставку на быстрое обновление парка ПК и увеличение производительности ПК с новыми микропроцессорами 486 и Pentium. Версия MathCAD PLUS 5.0, к примеру, явно ориентирована на быстрые ПК с этими процессорами. Она требует ОЗУ объемом не менее 8 Мб и занимает на жестком диске почти 16 Мб. Кроме того, нужно резервное пространство на диске почти такого же размера для записи временных файлов, возникающих при использовании механизма виртуальной памяти Windows.

Тем более оказалось приятной неожиданностью то, что MathCAD PLUS 5.0 не только спокойно «установился» на ПК 386DX автора (разумеется, с расширенным до 8 Мб ОЗУ), но и проявил заметно большую резвость, чем его куда более простые предшественники со скромными запросами на 2-4 Мб ОЗУ. Это сразу стало заметным по «мгновенному» появлению горизонтальных линеек пиктограмм и явному ускорению работы со встроенной базой помощи. Складывается впечатление, что в 2-3 раза повысилась и скорость выполнения вычислений и, что особенно ценно, построения сложных графиков — например, цветных контурных и 3D-поверхностей.

После инсталляции системы, занимающей, кстати, пять дисков формата 1,44 Мб (MathCAD 3.0 размещалась на трех дисках 1,2 Мб), в окне оболочки Windows появляется группа пиктограмм с названием MathSoft Apps (рис. 1).

В этой группе имеется четыре пиктограммы:

Mathcad PLUS 5.0 — пуск системы Mathcad PLUS 5.0,

Mathcad Tutorial — пуск электронного самоучителя,

MathSoft Book Sampler — пуск электронной книги с примерами применения системы,

Desktop Reference — пуск электронного справочника с реферативной информацией.

Помимо возможностей, унаследованных от версии 5.0, версия PLUS 5.0 дает ряд новых: двухмерную нелинейную сплайн-аппроксимацию, использование функций, написанных пользователем на языке C++ (с поддержкой механизма DLL), обеспечение механизма динамической связи DDE при работе с векторами и матрицами, решение систем нелинейных дифференциальных уравнений, вычисление собственных значений и векторов для матриц с комплексными элементами, арифметические вычисления с произвольной точностью (до 4000 значащих цифр), вычисление пределов функций, решение переопределенных систем линейных уравнений, выполнение прямых и обратных преобразований Фурье, Лапласа и Z-преобразований. Восстановлен статус-кво экспертной системы SmartMath, введенной в версию 4.0. Временно исчезнув в версии 5.0, эта система вновь появилась в версии PLUS 5.0.

Заметно возросло информационное обеспечение системы. Помимо прекрасной и быстрой помощи, в систему встроены мультимедийный электронный учебник с динамической графикой, приличный набор пакетов применений и электронный справочник по математике, физике, электрорадиотехнике и другим отраслям науки и техники. Предусмотрена даже возможность разработки и применения электронных учебников, составляемых пользователями (их, кстати, можно готовить и на русском языке). Система стала не только мощным инструментом решения математических задач, но и вполне приличным справочником по математическим методам и расчетам в различныхобластях.

Электронный самоучитель — шаг к мультимедиа?

Электронный самоучитель в MathCAD PLUS 5.0 существенно переработан по сравнению с версией 4.0. В него введены игровые компоненты и средства динамической графики. При запуске учебника пиктограммой «Mathcad Tutotial» появляется преподаватель — джентльмен довольно комичной наружности, готовый дать вам необходимые пояснения. На первый раз он указывает на махонькое меню в правом нижнем углу окна самоучителя (рис. 2).

Дабы джентльмену было не скучно в его высокой преподавательской миссии, в помощь ему появляется бестолковая взбалмошная ассистентка, закидывающая своего шефа вопросами. Все действия в форме диалога между джентльменом и его подопечной развиваются на фоне окна системы Mathcad PLUS 5.0. И это исключительно важно — в системе реализован принцип объединения учебника с оболочкой системы!

Итак, по мере знакомства с системой джентльмен и его подружка перемещаются по всему окну и указывают (рукой или указкой) на необходимые действия. Рис. 3, например, показывает, как научиться задавать построение графика математической функции.

В ходе демонстрации возможностей системы джентльмен-преподаватель последовательно выполняет те действия, которые должен выполнять пользователь, а последний тут же наглядно наблюдает за их исполнением. Наш учитель легко перемещается по окну вправо и влево, вверх и вниз, ловко манипулируя всеми элементами пользовательского интерфейса системы. Если бы не отсутствие звука, то самоучитель вполне можно было бы отнести к превосходному продукту мультимедиа. Остается надеяться, что в следующей версии средства мультимедиа будут использованы на всю катушку!

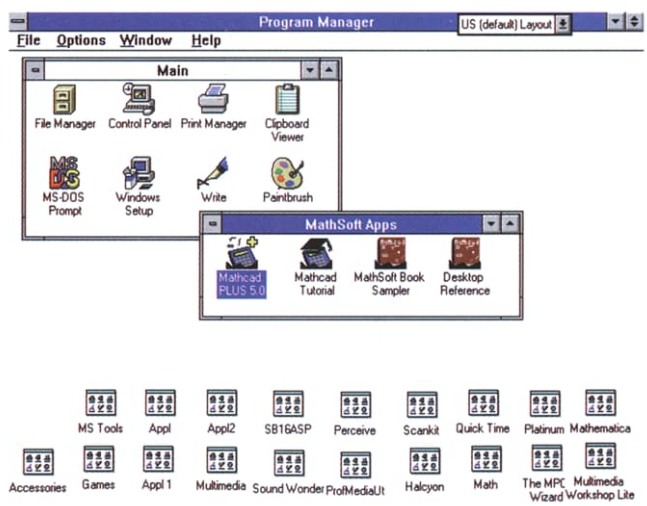


Рис. 1. Вид окна менеджера программ с программной группой системы MathCAD PLUS 5.0

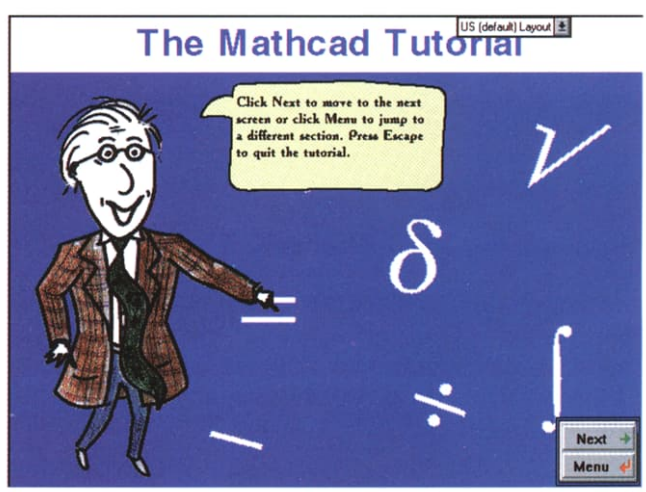


Рис. 2. Окно самоучителя в начале загрузки

От пакетов применений к электронным книгам!

Итак, самоучитель дает весьма полезную информацию тем, кто впервые намерен освоить систему. Однако многие пользователи (и их, пожалуй, большинство) уже знают систему MathCAD по ее предшествующим версиям 3.0 и 4.0. Для них наиболее полезными были бы пакеты применений системы, решающие различные задачи в реальной обстановке общения с системой.

MathSoft предприняла новые шаги в интеграции пакетов применения не только с оболочкой системы, но и со встроенными в нее двумя книгами. Одна из книг «MathSoft Book Sampler» является интерактивным описанием множества примеров применения системы в различных отраслях науки и техники — математике, физике, электрорадиотехнике и других. На рис. 4 показан титульный лист этой книги после ее загрузки.

Эта книга является одновременно сборником пакетов применений системы MathCAD PLUS 5.0 и описанием как общих, так и самых утонченных методов применения системы. Важно, что речь идет не только об описании примеров, но и о возможности тут же исполнить их в реальных условиях применения системы и при необходимости задать новые исходные данные или даже подкорректировать алгоритмы решения. Для этого вовсе не надо возвращаться из книги в оболочку системы и по-новому вводить данные — книга реализована внутри оболочки! Пользователь имеет возможность тут же опробовать любой пример с другими, скорректированными им, исходными данными.

Рис. 5 показывает оглавление книги, которое поможет вам сделать вывод о классах задач, рассматриваемых в данной книге.

Уже отмечалось, что в версию MathCAD PLUS 5.0 наконец-то введен решатель дифференциальных уравнений и систем с такими уравнениями в численном виде. Эту возможность трудно переоценить, так как многие научно-технические задачи (особенно относящиеся к анализу динамических систем и к их математическому моделированию) базируются на численных методах решения систем дифференциальных уравнений. В прежних версиях MathCAD пользователю приходилось самостоятельно составлять документы с решением таких уравнений, например методом Рунге-Кутта, что было делом не очень простым. К тому же решение загромождало текст документа. Теперь эта проблема решена, и довольно изящно.

Пример на рис. 6 из электронной книги «MathSoft Book Sampler» иллюстрирует технику решения дифференциального уравнения второго порядка. Виден и результат решения — в виде графика полученной временной зависимости.

Итак, по существу, вместо пакетов отдельных применений фирма MathSoft перешла к поставке с системой MathCAD электронных книг. В них применения системы хорошо документированы и сопровождаются реальными решениями конкретных задач. В системе помощи сообщается, что

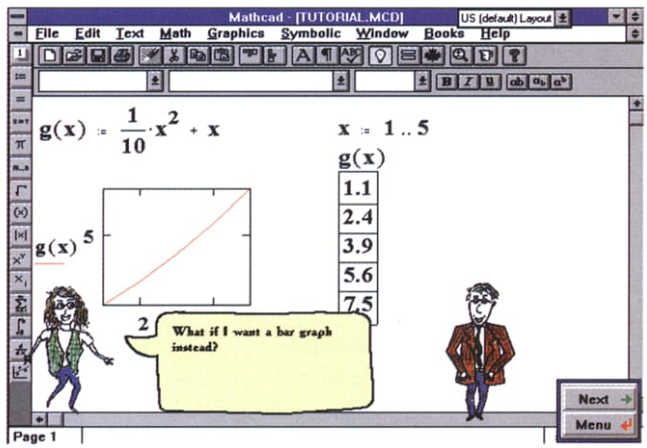


Рис. 3. Обучение построению графика функции

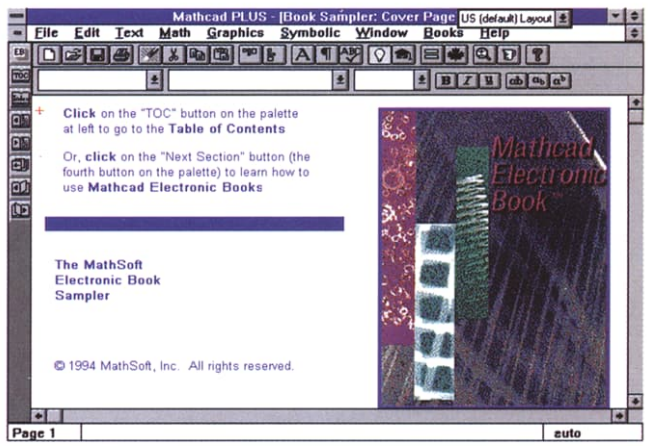


Рис. 4. Титульный лист книги «MathSoft Book Sampler»

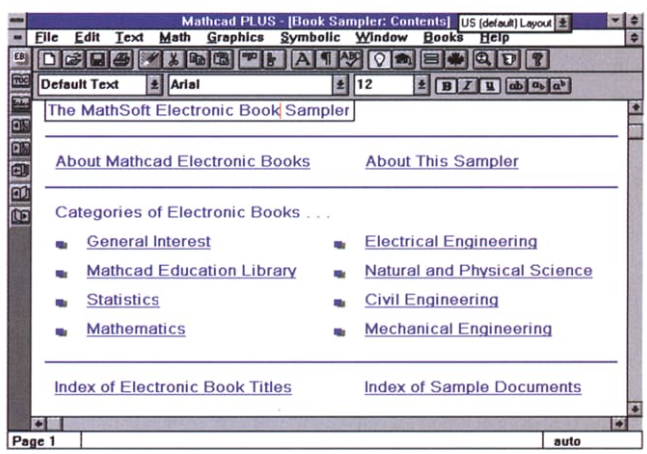


Рис. 5. Оглавление электронной книги с примерами решения задач

в 1994 г. пользователи уже могли приобрести 27 электронных книг! Специалист почти в любой области науки и техники может вполне самостоятельно составить такую книгу, включив в нее решение наиболее важных для него задач.

Встроенный справочник

Начиная с версии MathCAD 4.0 фирма MathSoft стала вводить в систему встроенный справочник. Он реализован также в виде электронной книги «Desktop Reference». В этой книге можно найти множество полезных для повседневной работы с системой справочных данных — математические формулы и методы вычислений, физико-химические свойства веществ, графики функций, изображения геометрических фигур и электрорадиосхем и так далее. Так что необходимость рыться в обычных справочниках и энциклопедиях если и не устраняется полностью, то во всяком случае сводится к минимуму.

ПРОГРАММЫ

MathCAD PLUS 5.0 заслуживает оценки «пять с плюсом» как самая мощная математическая система этого класса. Два десятка расширений и превосходный интерфейс отличают ее от популярной и поныне системы MathCAD 3.0 под Windows.

Рис. 7 иллюстрирует использование электронного справочника — выводится рисунок усеченного конуса и приводятся формулы для его поверхности и объема.

Встроенный справочник создан вовсе не только для созерцания формул и рисунков. Любую группу формул или рисунков можно выделить «мышкой», установив ее курсор в стороне от них, нажав левую клавишу и перемещая курсор в область нужных формул и рисунков. При этом они выделяются пунктирными прямоугольниками — точно так же, как блоки на экране редактирования документов. Нажав, как обычно, клавишу F3, можно выделенные элементы поместить в специальный буфер Clipboard.

Теперь, даже очистив экран от справочника, можно расположить выбранные объекты (формулы, надписи, рисунки) в месте, указанном графическим курсором системы. Для этого есть две возможности. Одна — нажать клавишу F4. Другая, присущая всем программам для Windows, — использовать команду Paste в позиции Edit редактора документов.

Таким образом, помещенную в электронный справочник информацию можно использовать в текстах документов, составляемых пользователем для решения своих прикладных задач. Кстати, подобным образом (используя средства редактирования документов и обмен через буфер) можно изменять содержимое справочника и вносить в него новые разделы.

Быстрая помощь

Работа с электронными книгами выглядит вполне сносно, хотя начальная загрузка их ввиду большого объема информации занимает немало времени. Зато база данных оперативной помощи доработана с блеском — вызов меню помощи и доступ к справке обеспечиваются почти мгновенно! Рис. 8 продолжает тему решения дифференциальных уравнений — на нем показан раздел помощи, посвященный этой проблеме.

Выбрав нужную позицию меню, можно получить довольно подробную справку об интересующей вас проблеме. К сожалению, формулы и текст из помощи нельзя перенести в окно редактирования документов. Это, пожалуй, единственный недостаток, кото-

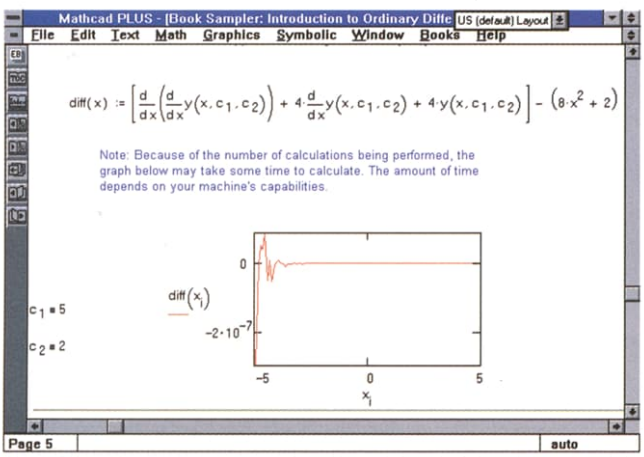


Рис. 6. Пример решения дифференциального уравнения, приведенный в электронной книге

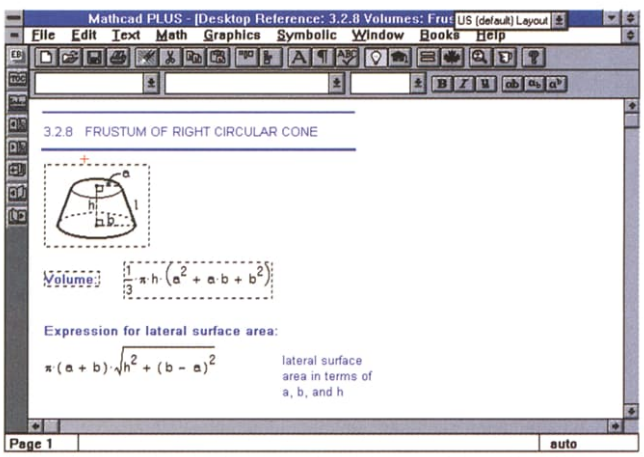


Рис. 7. Пример работы со встроенным справочником

рый удалось подметить среди справочного великолепия системы.

Улучшенные возможности подготовки документов

Начиная с версии 3.0 в системе MathCAD предусмотрен ввод большинства операторов по выбору из меню. Меню операторов в виде переключаемых столбцов пиктограмм расположено слева от основного окна редактирования документов. Такое решение позволило, наконец, избавиться от необходимости запоминания комбинаций клавиш, с помощью которых вводятся операторы — например, знак присваивания «:=» или знак интеграла. Осталась одна проблема: при введении функций надо помнить, как они записываются, да и вообще — какие из них существуют.

Уже в версии 4.0 был заложен ввод функций по выбору. Это «излишество» было устранено в версии 5.0, но реанимировано в версии PLUS 5.0. Желаемая функция может быть введена по месту расположения курсора, будучи выбранной из списка, который содержится в окне выбора функций. Это окно появится на экране, если в позиции Math главного меню выбрать команду Insert Function... (рис. 9).

Возможность выбора из списка устраняет синтаксические ошибки при вводе функции в текст документа. Кроме того, можно полистать список функций и наверняка найти в нем функции, о которых вы, скорее всего, даже не слышали.

Еще одно нововведение — синтаксический контроль математических и научно-технических терминов с помощью встроенной системы Check Spelling. При ее вызове появляется диалоговое окно (рис. 10), в котором отображаются термины, вызвавшие у системы контроля подозрения, и предлагаются варианты для замены.

Увы, все русскоязычные научно-технические термины система контроля принимает за ошибочные, а в качестве альтернативы — просто дает их повторение! Так что для наших пользователей польза от этой системы видна лишь в одном случае — если они работают на «них» и пишут свои работы на английском языке.

Хороший документ должен радовать своим опрятным внешним видом. И здесь появилось новшество — обзор страницы документа Preview перед выводом ее на печать. Он реализуется командой Print Preview в позиции File главного меню. При этом в специальном окне обзора можно просмотреть макет страницы и решить вопрос об оптимальном расположении на странице текстовых комментариев, формул и рисунков (рис. 11).

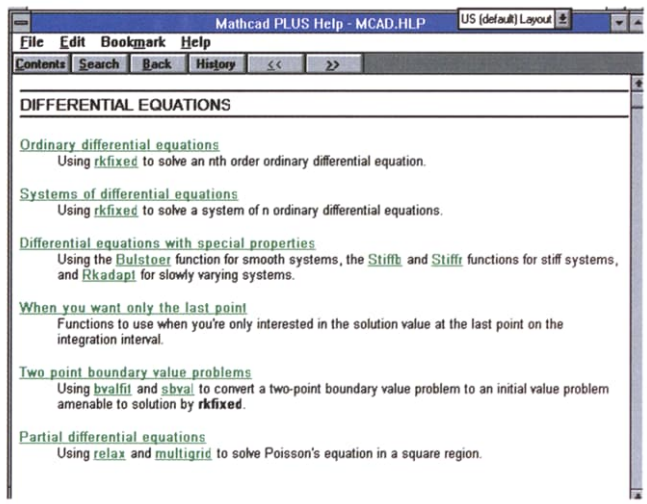


Рис. 8. Меню системы помощи, относящееся к решению дифференциальных уравнений

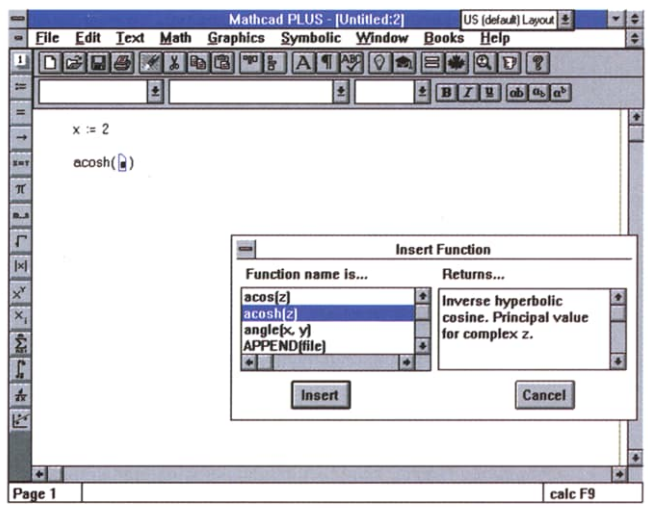


Рис. 9. Включение функции из списка

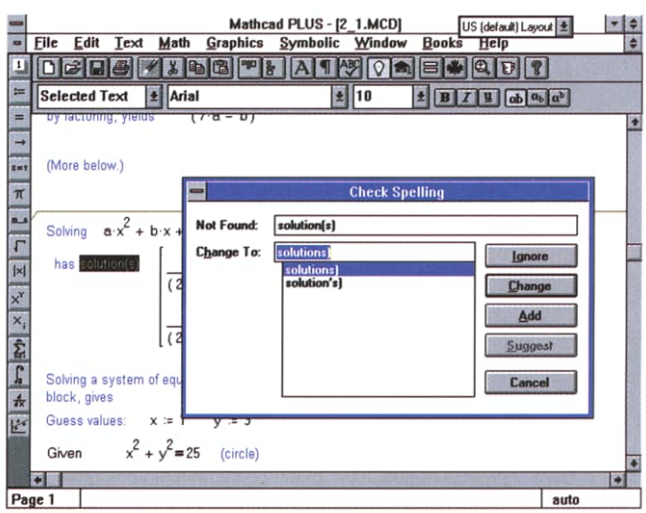


Рис. 10. Контроль синтаксиса математических и научно-технических терминов

Если документ большой и не помещается на странице, то можно с помощью указателей скроллинга страницы обеспечить оптимальное размещение на странице нужных объектов документа. После этого можно распечатать страницу принтером.

Девочки у нас не дуры — могут вмиг решить дифуры!

Из новых математических возможностей версии PLUS 5.0 самая мощная и привлекательная — это возможность решения численными методами дифференциальных уравнений и систем с ними — в студенческой среде именуемых «дифурами». Как всегда в MathCAD эта возможность доведена до верха совершенства — стоит просто записать систему, указать перед этим начальные условия и использовать одну из многих функций, порождающих решение в виде матрицы значений независимой переменной и значений производных. Эту матрицу тут же можно представить графически.

Отметим две из таких новых функций:

rkfixed(X,a,b,n,D) и Rkadapt(X,a,b,n,D).

Здесь: X — вектор искомых переменных, a и b — нижнее и верхнее значения независимой переменной (например, времени), n — число точек решения и D — имя функции, задающей систему дифференциальных уравнений.

Рис. 12 показывает решение системы из двух нелинейных дифференциальных уравнений с помощью функции rkfixed, реализующей обычный метод Рунге-Кутта с фиксированным шагом решения (b−a)/n. Этот метод дает хорошие результаты, если решения не содержат резких скачков.

Решение на рис. 12 иллюстрируется графиком, известным как фазовый портрет колебаний.

Другой рисунок (рис. 13) показывает то же решение, но полученное с помощью функции Rkadapt, реализующей адаптивный метод Рунге-Кутта. При этом шаг решения подстраивается под крутизну зависимостей, описывающих решение.

На этот раз решение представлено графиками временных зависимостей, которые позволяют сделать вывод о том, что решение описывает затухающие почти синусоидальные колебания. Помимо указанных функций версия PLUS 5.0 содержит и ряд других для решения дифференциальных уравнений специального вида.

Известно, что большинство нелинейных дифференциальных уравнений и систем не имеет аналитических (то есть символьных) решений и может решаться только численными методами. Решатель дифференциальных уравнений, введенный в версию PLUS 5.0, несомненно делает ее особо привлекательной для инженеров и научных работников, имеющих дело с такими уравнениями.

Отменная графика

Графика MathCAD PLUS 5.0 просто великолепна! В этом нетрудно убедиться, рассматривая рис. 14, на котором представлено построение

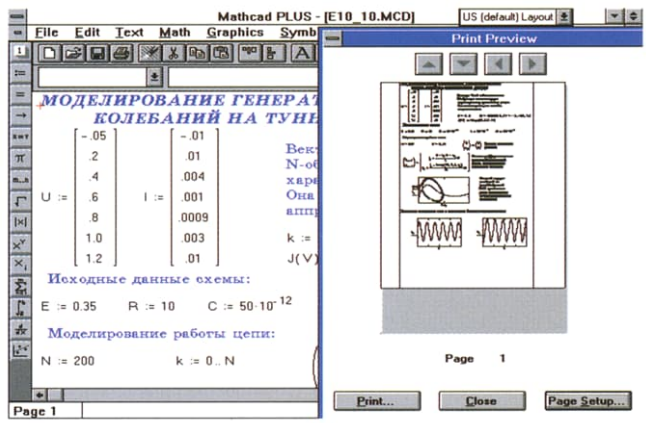


Рис. 11. Наблюдение макета набранной страницы документа

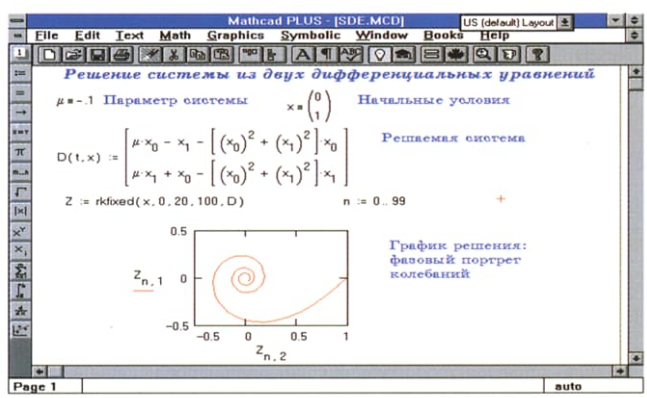


Рис. 12. Решение системы дифференциальных уравнений методом Рунге-Кутта с фиксированным шагом

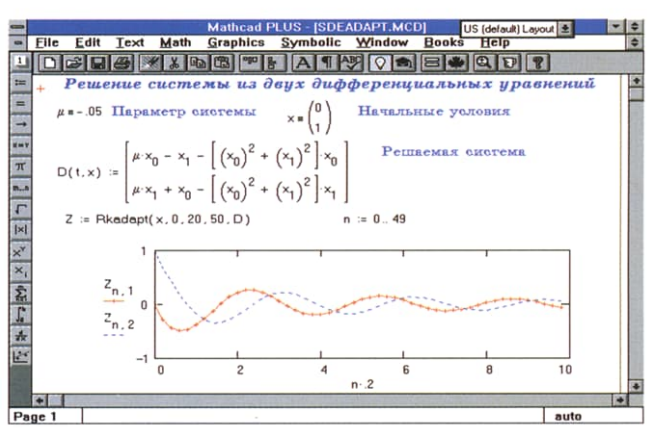


Рис. 13. Решение системы дифференциальных уравнений адаптивным методом Рунге-Кутта

пика. Поверхность представлена одновременно двумя графиками — в трехмерном пространстве и в виде контурной диаграммы с линиями равных высот.

По эффективности функциональной цветной закраски MathCAD стала соперничать с Mathematica 2 — системой, у которой графика всегда была козырным тузом. Впрочем, такой она и осталась — многие виды специальных графиков MathCAD по-прежнему не строит. К примеру, нельзя строить графики нескольких пересекающихся в пространстве фигур.

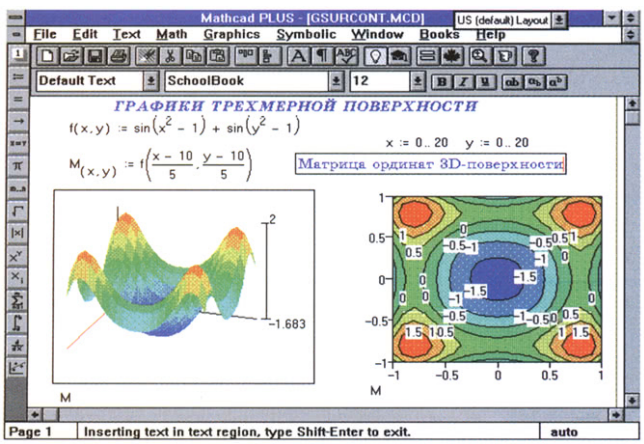


Рис. 14. Построение графиков трехмерной поверхности

О ложке дегтя в бочке с медом

В целом, несмотря на существенные дополнения (их более 20 в сравнении с версией 3.0) MathCAD PLUS 5.0 осталась математической системой для всех! Разумеется, этой системе по-прежнему далеко до лидеров Maple V и Mathematica 2 (особенно последней версии 2.2.2) в части реализации задач символьной математики и математической графики. Возможность численного решения дифференциальных уравнений несомненно повысит интерес к MathCAD PLUS 5.0 со стороны многих пользователей — ученых, инженеров и студентов. Как обычно, выше всех похвал оценка пользовательского интерфейса системы. К сожалению, цена за эти удовольствия велика — стоимость MathCAD PLUS 5.0 подскочила с 295 долларов США (за упрощенную версию 5.0) до 595 долларов (без стоимости расходов на упаковку и пересылку и таможенных пошлин).

Из принципиальных недостатков следует отметить определенные проблемы с инсталляцией в случае, если на вашем ПК стояла ранняя версия системы. Есть проблемы и с обеспечением нормальной работы с принтерами, особенно струйными фирмы «Hewlett-Packard». В используемой автором системе так и не удалось добиться распечатки документов принтером. Впрочем, такое было и с версией 4.0 — видимо, есть некорректность в вызове драйверов печати в 32-разрядном расширении Windows, используемом версиями MathCAD 4.0 и выше. Это уж действительно ложка дегтя в бочке с медом!

Если вы лишены возможности решить эти проблемы, обратившись за помощью к фирме «MathSoft Inc.», то вам придется довольствоваться копированием документов в буфер Clipboard и распечаткой их (возможно, по частям) с помощью других подходящих программ. В версии 3.0 таких проблем совсем нет (она не использует 32-разрядное расширение Windows). К тому же она требует вчетверо меньше объема ОЗУ для своей работы — 2 Мб. Так что нашим пользователям ПК 386/486 с ОЗУ 2-4 Мб можно не спешить с применением MathCAD PLUS 5.0. Тем более, что на пороге появление версии 6.0 — возможно, она порадует нас новыми возможностями и полным отсутствием проблем с ее применением! К этому времени стоит расширить ОЗУ ПК до 8 или 16 Мб. ■

MathCAD 5.0

Спустя всего лишь год после создания версии MathCAD 4.0 фирма «MathSoft Inc.» выпустила версию 5.0, а затем и более совершенную версию PLUS 5.0. Поскольку последней посвящена статья, здесь отразим особенности промежуточной версии 5.0.

По сравнению с версией 4.0 в версию 5.0 добавлено не так уж и много: возможность создания собственных учебников пользователя, поддержка технологии DDE для векторов и матриц, создание колонтитулов на страницах документов и проверка синтаксиса с помощью системы Spell Checker. Улучшились условия подготовки документов — стал возможен ввод функций из списка и просмотр макета страницы документа (Preview).

Довольно неожиданным является отказ от экспертной и оптимизирующей системы SmartMath, широко разрекламированной в версии 4.0. Исчезла и возможность весьма удобного выбора формул из списка. По-видимому, эти ограничения связаны со стремлением фирмы-разработчика сохранить умеренные требования к аппаратным ресурсам ПК (ОЗУ минимальной емкости 4 Мб, 14 Мб на жестком диске) и скорости работы с системой.

Складывается впечатление, что фирма «MathSoft» решила создать две сильно различающиеся пятые версии из чисто экономических соображений. Версия 5.0 (без PLUS) стала стоить на сотню долларов дешевле, чем предшествующие версии — «всего» 295 долларов США. Те, кто уже имеют установленные на ПК версии MathCAD 3.0/4.0, вряд ли с восторгом оценят версию 5.0. Уж коли обновлять математику на своем ПК, то лучше сразу перейти к более мощной версии PLUS 5.0. Впрочем, в статье высказано и иное мнение.

Литература

1. **В. П. Дьяконов. Система MathCAD. Справочник.– М.: Радио и связь, 1993.**
2. **В. Дьяконов. Как выбрать математическую систему// Монитор-Аспект.–1993.– № 2.**
3. **В. Дьяконов. MathCAD прорывается в Windows// Монитор-Аспект.– 1993.– № 2.**

Сети нового поколения: технология FDDI

Р. Убайдуллаев

Технология FDDI (Fiber Distribuded Data Interface) — распределенный интерфейс данных на оптическом волокне — одна из лучших скоростных сетевых технологий, прошедшая испытание временем, ориентированная на использование оптического волокна в качестве среды связи. Благодаря этому сеть FDDI невосприимчива к помехам и обладает повышенной надежностью, защищенностью от несанкционированного доступа, большой пропускной способностью и многими другими преимуществами.

Введение

Скоростные вычислительные сети все сильней начинают проникать в нашу жизнь. Благодаря техническому прогрессу компьютеры становятся более мощными, а обработка и пересылка информации с их помощью — более быстрой и эффективной.

Другие причины также способствуют глубокому проникновению новейших сетевых технологий в наш мир. Во-первых, современные системы связи, в особенности распределенные банки данных и средства мультимедиа, остро ставят вопрос построения линий связи, обеспечивающих быструю и надежную передачу информации. Сюда можно отнести непрерывную передачу аудио- и видеоизображения, передачу огромных баз данных и графических файлов, достигающих нескольких десятков мегабайтов.

Во-вторых, это процесс интеграции предприятий и — как следствие — потребность в выделенной линии связи между банками, офисами, филиалами, удаленными на расстояние от нескольких до десятков километров. Сегодня в России многие крупные предприятия имеют несколько разрозненных сетей, базирующихся в основном на стандарте Ethernet, и нуждаются в их объединении. Причем часто оказывается необходимым связать между собой вычислительные сети (например, Ethernet, Token Ring), на которых уже работают разные и необязательно стандартизированные приложения, или установить быструю линию связи с несколькими серверами на базе различных компьютерных платформ. Если к этому добавить требование надежного функционирования всей сети, то получается непростая задача. ЛВС, построенная по стандарту FDDI, предназначена для ее решения.

Статья печатается на коммерческой основе.

Основные характеристики

Главная характеристика любой сети — это скорость передачи данных, или иначе полоса пропускания. У FDDI этот параметр достигает **100 Мбит/с** и существенно превосходит соответствующие значения 10 Мбит/с у сетей Ethernet или 4 и 16 Мбит/с у сетей Token Ring. Основные параметры этих сетей отражены в таблице 1.

По скорости передачи с FDDI сравнимы такие современные сетевые технологии, как стомегабитовая Fast Ethernet, существующая в двух основных стандартах: 100 Base-T и 100VG AnyLan. Их разработка завершилась только в середине 1994 года. Поэтому на данный момент нет достаточной практической проверки надежности сетевого оборудования Fast Ethernet. Однако с уверенностью можно сказать, что при использовании в качестве протяженных магистралей (10 км и более) сетям FDDI не составят конкуренцию сети Fast Ethernet, ограниченные расстоянием 3 км в обоих стандартах: 100 Base-T и 100VG AnyLan.

Другой «конкурент» — это ATM (Asynchronous Transfer Mode), режим асинхронной передачи. ATM, использующий технологию быстрого переключения каналов, призван работать в диапазоне от 22 до 622 Мбит/с. Разработка стандарта ATM в настоящее время практически завершена. Однако существующее оборудование пока очень дорогое.

Базовый стандарт FDDI основан на использовании оптико-волоконной среды для связи между сетевыми устройствами. Первоначальная версия использовала многомодовое оптическое волокно и светоизлучающие диоды, работающие на длине волны 1,3 мкм при расстоянии между станциями до 2 км.

В настоящее время в качестве среды передачи для FDDI также применяется одномодовое оптическое волокно и витая пара (на медном кабеле). Витая пара представлена в двух модификациях: экранированная витая пара (Shielded Twisted Pair — STP) IBM Type 1 и неэкранированная витая пара (Unshielded Twisted Pair — UTP) категории 5. При этом скорость передачи та же (100 Мбит/с), а допустимые расстояния между станциями:

для одномодового волокна — 50 км;
для STP и UTP — 100 м.

Таким образом, у пользователей появился выбор. Приведем сравнительный анализ оптического волокна и витой пары.

Преимущества волокна

Главные преимущества функционирования сети на оптико-волоконном кабеле следующие:

1) широкая полоса пропускания;
2) большие расстояния между станциями;
3) высокая помехозащищенность;
4) высокая защищенность от несанкционированного доступа;
5) гальваническая развязка элементов сети;
6) взрыво- и пожаробезопасность.

Широкая полоса пропускания. Градиентное многомодовое волокно имеет

Сравнительные характеристики FDDI, Ethernet и Token Ring *Таблица 1*

	FDDI	IEEE 802.3 Ethernet	IEEE 802.5 IBM Token Ring
Скорость передачи	100 Мбит/с	10 Мбит/с	4/16 Мбит/с
Число станций	500	1024	256
Максимальное расстояние между станциями	2 км (MMF) 50 км (SMF) 100 м (UTP кат. 5)	2,8 км (MMF) 500 м (10Base5) 185 м (10Base2) 100 м (10BaseT)	3 км (MMF) 150 м (IBM тип 1) 100 м (UTP кат.4) (16 Мбит/с)
Максимальная протяженность сети	100 км	2,8 км	Зависит от конфигурации
Среда	Оптическое волокно, витая пара	Оптическое волокно, витая пара, коаксиальный кабель	Витая пара, оптическое волокно

MMF (multy-mode fiber) — многомодовое волокно

SMF (single-mode fiber) — одномодовое волокно

полосу пропускания 500 МГц·км. Это означает, что его полоса пропускания на расстоянии 100 м будет 5 ГГц ($5 \cdot 10^9$ Гц), что превышает более чем на порядок аналогичную величину для витой пары (UTP категории 5). Еще больше полоса пропускания одномодового волокна — 5000 МГц·км. Широкая полоса пропускания допускает наращивание. Можно, к примеру, поставить оборудование FDDI там, где раньше использовались оптические трансиверы Ethernet, совершенно не меняя кабельных коммуникаций.

Большие расстояния между станциями. Одно из преимуществ оптико-волоконной среды — это слабое затухание и небольшая дисперсия распространяемого по волокну сигнала. По стандарту FDDI расстояние между станциями не превосходит 2 км, если применяется многомодовое волокно в качестве среды связи между станциями и светоизлучательные диоды на уровнях PHY. Использование одномодового оптического волокна и лазерных передатчиков позволяет увеличить расстояние между станциями до 50 км.

Высокая помехозащищенность. Второе преимущество волокна — его нечувствительность к электромагнитным помехам. На импульсы света, распространяющиеся по волокну, не воздействует электромагнитное излучение. Это делает волокно более надежным по сравнению с медным кабелем в случае его прокладки вблизи электродвигателей и другого излучающего оборудования. Такое излучение, воздействуя на электроны в проводнике, может приводить к сильным помехам на сетевых устройствах.

Высокая защищенность от несанкционированного доступа. Поскольку оптико-волоконный кабель практически не излучает сигналов в радиодиапазоне, то передаваемую по нему информацию трудно подслушать. Это свойство особенно важно при создании линий связи для правительственных, банковских и некоторых других специальных служб, предъявляющих повышенные требования к защите данных.

Гальваническая развязка элементов сети. Пятое преимущество оптического волокна заключается в его изолирующем свойстве. Волокно помогает избежать электрических земельных петель, которые могут возникать, когда два сетевых устройства неизолированной локальной вычислительной сети, связанные медным кабелем, имеют заземления в разных точках здания, например на разных этажах. При этом может возникнуть большая разность потенциалов, и электрический ток способен разрушить сетевое оборудование. Для волокна этой проблемы просто нет.

Взрыво- и пожаробезопасность. Из-за отсутствия искрообразования оптическое волокно повышает безопасность сети на химических, нефтеперерабатывающих предприятиях, при обслуживании технологических процессов повышенного риска.

Преимущества меди

Причины, по которым заказчик может предпочесть медный кабель оптико-волоконному, следующие:

1) низкая стоимость подключения;
2) удобство использования в небольших рабочих группах.

Низкая стоимость подключения. Витые пары STP IBM тип 1 и UTP категории 5 могут существенно уменьшить затраты на сетевое оборудование, так как они не требуют установки дорогостоящих оптических приемо-передатчиков и пассивных компонентов волоконной оптики.

Удобство использования в небольших рабочих группах. Витая пара будет удобной при использовании концентратора в малых рабочих группах, в конструкторских бюро, хотя это удобство скорее является следствием меньшей стоимости FDDI-концентратора с выходами для подключения витых пар.

Топология FDDI

У любой сети есть как физическая, так и логическая топология.

Физическая топология показывает естественное размещение сетевых станций, а также кабельной системы, посредством которой устанавливается физическая связь между станциями.

Логическая топология дает представление о пути, по которому двигаются данные от станции к станции.

Если логическая топология сети FDDI всегда есть кольцо, то физическая топология может быть весьма разнообразной. Оптическое волокно, плохо адаптируемое для использования в качестве общей шины, подходит для конфигурации «точка-точка» между станциями. Таким же свойством обладает и витая пара. Наиболее общая физическая топология — двойное кольцо деревьев (рис. 1а). В этой топологии сеть образована иерархическими деревьями, в корнях которых находятся концентраторы, непосредственно подключенные к двойному кольцу (Dual-Attachment Concentrator — DAC). В кольцо могут быть также подключены рабочие станции (Dual-Attachment Station — DAS). Физическое кольцо образуется соединениями «точка-точка» между DAS и DAC, причем каждое такое соединение осуществляется по паре оптических волокон, где свет распространяется в разных направлениях (одно из колец резервное). На рис. 1б приведена соответствующая логическая топология в простейшем случае, когда концентраторы не выполняют роль мостов и маршрутизаторов.

Корневые концентраторы образуют кольцевую магистраль FDDI и, таким образом, являются ключевыми. К этим концентраторам подсоединяются концентраторы с простым подключением (Single-Attachment Concentrator — SAC) или станции с простым подключением (Single-Attachment Station — SAS).

Двойное кольцо деревьев — это наиболее универсальная и гибкая топология, охватывающая все преимущества технологии FDDI. Двойное кольцо деревьев позволяет создать огромную и сложную сеть, которая благодаря резервному кольцу будет сохранять целостность при повреждении линии связи кольца или при отключении станции или концентратора от кольца. Допол-

нительно для повышения надежности возможно подключение концентраторов или станций к кольцу через специальное устройство — оптический обходной переключатель (Optical Bypass Switch).

Двойное кольцо деревьев может естественным образом вырождаться в следующие три варианта:

- двойное кольцо;
- отдельный концентратор;
- дерево концентраторов.

Двойное кольцо — топология, объединяющая только станции с двойным подключением. Она удобна и наиболее часто применяется в тех случаях, когда есть небольшое число станций DAS, которые следует объединить в сеть, и нет необходимости прибегать к дорогостоящим концентраторам.

Отдельный концентратор — это концентратор, который не подключается к двойному кольцу и имеет внутреннюю шину FDDI (backplane). К нему подключаются станции SAS. При такой топологии администратору сети легко контролировать сетевые устройства конечных пользователей и ограничивать доступ пользователей к сетевым ресурсам, используя функции концентратора. При выходе из строя или выключении станции соответствующий концентратор автоматически отключит ее от сети.

Дерево концентраторов изменяет предыдущий вариант для случая иерархического подключения. Эта топология дает эффективное решение, когда нужно объединить большое количество станций в пределах одного здания или в пределах одного этажа здания.

Принцип действия

Работа сети FDDI основана на детерминированном маркерном доступе к логическому кольцу. До тех пор пока нет передачи данных, от станции к станции циркулирует (имеется в виду логическое кольцо) единственный маркер — укороченный пакет служебных данных, при получении которого станция обретает возможность передавать информацию. На рис. 16 квадратами обозначены места взаимодействия станций с сетью. Если станция не хочет передавать, то, получив маркер, она пропускает его дальше по кольцу. При желании передавать данные станция, получив маркер, удерживает его до тех пор, пока ведет передачу. Данные организуются в пакеты длиной до 4500 байтов (кадры). В любой момент времени передавать может только одна станция, а именно та, которая захватила маркер. После окончания передачи станция сразу же высвобождает маркер — посылает его вслед отправленным кадрам.

Каждая станция сети читает адресные поля получаемых кадров. В том случае, когда собственный адрес станции отличен от поля адреса получателя, станция просто пропускает кадр дальше по кольцу. Если же собственный адрес станции совпадает с полем адреса получателя, станция считывает для себя данные из этого кадра, проставляет в специальном поле кадра бит «принято» и отправляет его дальше по кольцу. Когда кадр возвращается обратно к станции-отправителю, она ликвидирует его.

Маркерный доступ — это одно из наиболее эффективных решений. Благодаря этому реальная производительность кольца FDDI при большой загруженности достигает 95%. Для примера, производительность сети Ethernet с ростом загруженности достигает 30% от пропускной способности.

Кабельная система

Связь «точка-точка» между станциями в кольце не только упростила стандартизацию, но также позволила разным участкам кольца иметь свои особенности. Например, один участок кольца может использовать одномодовое волокно, другой — многомодовое, третий — витую пару.

На практике сеть FDDI имеет смешанную кабельную систему, пример которой приведен на рис. 2. Для связи между зданиями или между этажами используются магистральные многоволоконные оптические кабели. По этажу разводка до концентратора осуществляется оптическим мини-кабелем. От концентратора к рабочей станции можно провести как оптический мини-кабель, так и витую пару.

Сетевое оборудование

Ведущими производителями сетевого оборудования FDDI являются фирмы «3Com», «Cabletron», «Fibronics», «Cisco» и другие.

АО «ВИМКОМ» предлагает оборудование FDDI фирм «3Com» и «Fibronics». На наш взгляд, оборудование фирмы «3Com» (Lanplex 5000/6000, LinkBuilder 3GH) хорошо подходит для организации центрального узла локальной сети масштаба предприятия, которая связана с другими сетями по стандарту FDDI, а оборудование «Fibronics» — для организации связи по FDDI сетей рабочих групп, которые требуют высококачественного доступа рабочих групп и отдельных станций к серверам. Для этого используются концентраторы FDDI и устройства Switch-Ethernet с выходом на FDDI.

В дальнейшем будут рассмотрены вопросы функционирования этого оборудования, а также более общие вопросы использования мостов и маршрутизаторов при объединении различных сетей магистралью FDDI. ■

КОНТАКТЫ

АО «ВИМКОМ»

Россия 111524 Москва,
ул. Электродная, 10,
телефон: 306-1763,
факс: 306-4101
E-Mail: rustam@INFA.msk.su

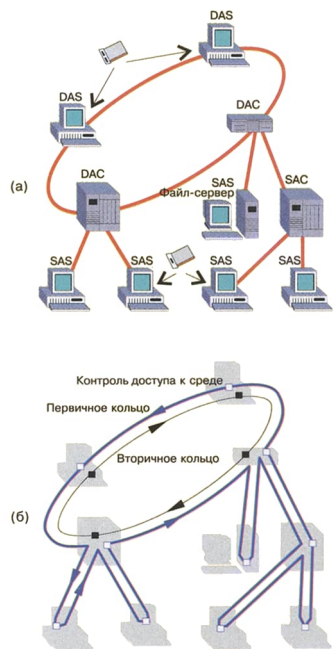


Рис. 1 (а) Физическая топология: двойное кольцо деревьев (б) Логическая топология: кольцо

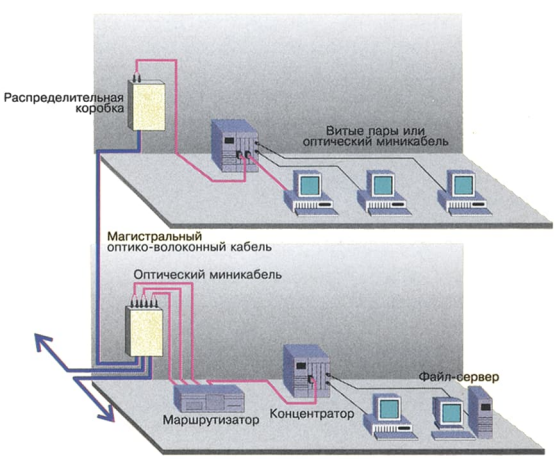


Рис. 2 Смешанная кабельная система FDDI

Вы могли бы и больше заплатить за сервер, но придется ограничиться меньшими расходами

ALPHA™ GENERATION

Новое 64-разрядное семейство AlphaServer сегодня, если Вам нужна высочайшая производительность 64-разрядных RISC-процессоров, непревзойденные возможности наращивания вычислительной мощности, исключительная надежность и лучшие гарантии для серверов в промышленности, Вы вполне можете получить это без чрезмерных затрат, если купите одну из удивительных моделей нашего нового семейства AlphaServer. Новые серверы поставляются с обширным программным обеспечением — свыше 6000 прикладных программ UNIX®, Windows NT™ и Open VMS™. Они предусматривают установку дополнительных аппаратных средств стандартной для промышленности локальной шины PCI, имеют увеличенную пропускную способность системы ввода-вывода, больше оперативной и больше внутренней массовой памяти. Фактически единственное, что не увеличено, это цена.

Никто и нигде не предложит Вам столь выгодной покупки!

	AlphaServer 1000 4/200	AlphaServer 2000 4/200	AlphaServer 2100 4/275
Число процессоров (SMP)	1	1 – 2	1 – 4
TPS (транзакция/с)	до 285	до 400	до 850
SPECint 92	135,8	126,7	200,1
Макс. показатель SPEC...92	3135	5778	15470
Число гнезд ВВ	2PCI,7EISA 1комбинир.	3PCI,1EISA	3PCI, 8EISA
Макс. пропускная способность ВВ	132 Мбайт/с	132 Мбайт/с	132 Мбайт/с
Макс. емкость ОЗУ	512 Мбайт	640 Мбайт	2 Гбайт
Макс. емкость внутренней массовой памяти	14 Гбайт	16 Гбайт	32 Гбайт

digital™

Представительства:
в Москве тел. (095) 244-9540
в С.-Петербурге тел. (812) 119-1040
в Киеве тел.: (044) 295-6910, 295-1431. Факс: (044) 293-6041

Это сладкое слово «надежность»

Прежде чем приобрести персональный компьютер, сервер или рабочую станцию, специалисты задумываются о том, что они хотели бы получить. Мощность, производительность, достаточную емкость внутренней и внешней памяти, скорость доступа к данным, гибкость адаптации к требованиям пользователя? Безусловно. Без преувеличений, техника корпорации «Digital» (DEC) соответствует названным параметрам, в чем читатель может убедиться, просмотрев статьи в «Мониторе» 7.94, 8.94 и 1.95, в которых рассказывается о направлениях деятельности и достижениях фирмы DEC.

Но есть еще одна особенность, о которой не забывают ни поставщики, ни покупатели вычислительной техники. Без нее все усилия производителя уйдут в песок, а мощный компьютер превратится в сломанную игрушку, ненужную железку с электронной начинкой. Эта магическая характеристика — **надежность**. Толковый словарь определяет это слово как «внушающий полное доверие, верный; прочный, крепкий». Итак, поговорим о новых разработках компании «Digital Equipment», обращая особое внимание на их соответствие критерию надежности.

Prioris — уникальная архитектура, позволяющая наращивать мощность

Корпорация «Digital» анонсировала семейство серверов PC — **Prioris HX** — с широкими возможностями наращивания. Машины Prioris HX представляют собой первые две системы нового класса — серверов симметричной мультипроцессорной обработки приложений на базе процессоров Pentium. Создан также продукт для удаленного управления этими серверами — **Remote Server Manager**.

Существующие серии персональных компьютеров DECpc MTE и DECpc XL будут теперь называться **Prioris MTE** и **Prioris XL**. Появилось и новое программное обеспечение для помощи начинающим в быстром освоении новых продуктов — «Getting Started». Таким образом, сейчас DEC имеет полный спектр многофункциональных серверов, настольных компьютеров и ПК-блокнотов.

Системы **Prioris HX** полностью оптимизированы для удовлетворения потребностей заказчиков за счет широких возможностей наращивания мощности процессора и объемов дисковой и оперативной памяти. Средство монтажа в стойке, используемое во всех серверах семейства Prioris, позволяет минимизировать площадь, занимаемую компьютерными информационными центрами.

Базовая конфигурация компьютера **Prioris HX 590** включает процессор Pentium 90 МГц, 16 Мб памяти и адаптер SCSI-2 Fast-Wide PCI. Сервер **Prioris HX** 590DP имеет два Pentium процессора по 90 МГц, 32 Мб памяти и адаптер SCSI-2 Fast-Wide PCI.

Уникальность архитектуры Digital осуществляется в конструкции «материнская плата/ дочерняя плата», при которой все компоненты процессора помещаются на дочернюю плату, а общесистемные средства — на материнскую. Такой модульный подход предоставляет прекрасную возможность перехода к следующим поколениям технологий на базе одного или нескольких процессоров Intel.

Prioris HX — это первый сервер PC, использующий мост PCI-PCI с шестью гнездами базовых шин PCI, а с шестью дополнительными гнездами EISA — всего двенадцать гнезд.

Высокая надежность

«Корпорация «Digital» производит промышленные серверы уже в течение 20 лет, — говорит Дуан Дикхат (Duane Dickhut), вице-президент подразделения серверов PC Digital. — Мы хорошо понимаем проблемы построения распределенных сетей и вкладываем весь годами накопленный опыт в надежность и качество новых серверов PC.»

Системы Prioris HX Digital — это серверы для рабочих групп в несколько сотен человек. С их помощью легко решается проблема объединения локальных сетей. Системы Prioris HX предлагают следующие средства обеспечения надежности и отказоустойчивости:

- семь панелей для устройств с «горячим» переключением и различными уровнями поддержки технологии дисковых массивов (Redundant Array of Inexpensive Disks, RAID);
- поддержка контроллера RAID Digital с одним или тремя каналами для внутренних или внешних интерфейсов SCSI;
- 512 кб кэш;
- поддержка памяти кода коррекции ошибок (Error Correction Code, ECC) до 256 Мб для защиты от разрушения данных;
- использование второго (дублирующего) источника питания на 450 ватт, поддерживающего автоматическое восстановление при сбоях;
- дублирование системы охлаждения с использованием отказоустойчивых вентиляторов;
- панель управления для оператора, отображающая статус системы;
- двойная канальная основа с «горячим» переключением;
- средство удаленного управления сервером.

Поскольку современные деловые приложения базируются на большом разнообразии сетевых и операционных систем, серверы **Prioris HX** поддерживают различные компьютерные платформы, включая Novell Netware 3.X и 4.X, Microsoft Windows, Windows for Workgroups, Windows NT Server 3.5, SCO UNIX, Novell UnixWare, IBM OS/2 SMP, Banyan Vines и PATHWORKS Digital.

Высокая производительность

Архитектура серверов **Prioris HX** построена так, чтобы оптимизировать производительность с помощью различных средств: контроллеров и устройств PCI и

SCSI-2, 512 Мб памяти и 512 кб кэш, поддержки 64-разрядной памяти для одного и двух процессоров, использования Pentium 90, а в дальнейшем — и процессоров Intel следующего поколения, поддержки расширенных средств высокопроизводительной шины PCI, в том числе PCI для Ethernet.

Пользователям обеспечен широкий диапазон контроллеров SCSI, включая одно— и многоканальные, базирующиеся на EISA или PCI. Десять внешних выходов SCSI совместно с внешними стойками памяти позволяют расширить дисковое пространство.

Удаленное управление серверами

Новый продукт фирмы DEC для удаленного управления серверами — **Remote Server Manager** — базируется на серверах EISA Digital. Используя независимый от конкретной топологии сети интерфейс протокола «точка-точка» (Point-to-Point Protocol, PPP), **Remote Server Manager** дает возможность управлять удаленными серверами и отслеживать их состояние по телефонным линиям с использованием модема.

Remote Server Manager не зависит от операционной системы и требует наличия всего одного пакета для контроля серверов с различными типами операционной среды. Управление удаленными станциями из одного центра использует многозадачные возможности Windows NT, при которых несколько сеансов отображаются одновременно на одной консоли управления.

С помощью консоли или дискеты можно загружать программное обеспечение или стартовать систему с дискеты по телефонной линии, при этом отпадает необходимость модификации программного обеспечения на местах и минимизируется количество вызовов системных служб. Вместо вызова системного персонала администратор может использовать средства удаленной диагностики, сохраняя тем самым время, затрачиваемое на визиты в удаленные точки сети.

Расширенные средства обеспечения секретности имеют градации паролей и ответную проверку вызовов и создают различные уровни секретности. Кроме того, установленные в этих серверах датчики могут предупреждать администраторов о степени внутреннего и внешнего напряжения и температуре внутри устройства.

DEC обеспечивает секретность Internet для CableNet'94

Компания «Digital» участвовала в выставке кабельного оборудования (Westwrn Cable Show) CableNet'94 в Калифорнии. Она не только успешно демонстрировала там свою продукцию в основной рубрике «Internet — по кабелю», но и предоставила средства обеспечения секретности Internet для всех экспонентов выставки. Выставка продемонстрировала тенденцию слияния кабельной и сетевой компьютерной технологии и показала продукты 45 компаний-участников.

Услуги по секретности в Internet (Internet Security Services) объединяют средство Digital Screening External Access Link (S.E.A.L.) и консультации по настраиваемым решениям. **S.E.A.L.** — это ряд компонентов технического и программного обеспечения, позволяющих контролировать доступ в секретных и незащищенных сетях.

Ну, а основной задачей фирмы на выставке CableNet'94 была демонстрация высокоскоростного доступа к сети Internet по телевизионным каналам.

Brouter Internet ChannelWorks обеспечивает скоростной доступ к Internet

Корпорация «Digital» представила **Brouter** (маршрутизатор и мост) **ChannelWorks** — уникальный продукт, включающий в себя возможности высокоскоростного доступа к Internet по стандартным кабельным телевизионным сетям. Продукт обеспечивает взаимодействие локальных сетей Ethernet по кабельным телевизионным каналам со скоростью до 10 Мб/с. Можно сказать, что **ChannelWorks** расширяет возможности Internet, так как большая скорость доступа избавляет от длительных задержек при передаче информации приложениями Internet. Это не только удобно при работе с интерактивными приложениями мультимедиа, но также экономически выгодно, поскольку транзакции Internet занимают меньшее время.

На выставке CableNet'94 были также продемонстрированы беспроводной доступ из офиса в сеть Internet по кабелю и высокоскоростной доступ к базам данных в режиме «on-line», включая доступ к Internet.

Беспроводной доступ к Internet

DEC обеспечивает беспроводной доступ к Internet с помощью сетевого решения, объединяющего **ChannelWorks** и **RoamAbout**, — серии продуктов для беспроводных локальных сетей. На одной из демонстраций на выставке CableNet'94 пользователь с ПК-блокнотом, перемещаясь по залу, мог управлять соединениями кабельной сети, включая доступ к Internet, точно так же, как он делал бы это, находясь за рабочим столом.

RoamAbout Access Point и сетевой адаптер **RoamAbout PCMCIA** — наиболее производительные беспроводные продукты на сегодняшнем рынке — предлагают широкие функциональные возможности, включая связь «точка-точка» между зданиями и расширенное управление SNMP. **RoamAbout Access Point** служит мостом между ПК-блокнотами, локальными сетями и Brouter Internet ChannelWorks. Он работает в сочетании с сетевым адаптером **RoamAbout PCMCIA** размером с кредитную карточку и радиомодемом, подсоединенным к ПК-блокноту.

Скоростной доступ к базам данных в режиме «on-line»

На выставке CableNet'94 компания DEC совместно с поставщиком «on-line» услуг — фирмой «America On-line» — демонстрировала высокоскоростной доступ к информационно насыщенным приложениям, включая Internet, по технологии кабельного телевидения. Было показано, что продукты **ChannelWorks** снимают ограничения скорости доступа, заменяя традиционные телефонные линии новыми средствами связи. При подключении **Brouter Internet ChannelWorks** к информационно насыщенным приложениям «America Online» было отмечено огромное увеличение производительности в получении всех услуг.

Доктор Лауренс Волкер (Laurence Walker), вице-президент и генеральный менеджер подразделения сетевых продуктов DEC, сообщил: «Представленные технологии демонстрируют постоянные усилия «Digital» "поднять уровень скорости" информационных потоков посредством улучшения взаимодействия с кабельными сетями. Мы сконцентрировали внимание на обеспечении высокоскоростного направления на этом пути, с учетом расширения средств безопасности для пользователей, хост-ЭВМ и сетей.» ■

Компьютерный «ларёк» 2.95

Код	Продукт	Краткая характеристика	Объем поставки	Цена, $	Требования к оборудованию
SL-01	Turbo Games 3.0	Библиотека для Турбо Паскаль 6.0-7.0. Спрайты, трехмерная графика, стереоэффекты. Доступ к ZIP файлам. Поставка в исходных текстах.	1 дискета + документация	10	IBM PC EGA/VGA
SL-02	Data Vision 2.5	**Объектно-ориентированная библиотека для Borland Pascal. Унифицированный интерфейс с БД различных форматов (dBase/Clipper, Paradox, редакторы БД, QBE, генератор отчетов).Поставка в исходных текстах.**	1 дискета + документация	110	IBM PC
SL-03	DVision DBF Interface 2.0	Поддержка DBF-файлов. Работа с MEMO-полями формата dBase, Clipper, FoxPro. Поставка в исходных текстах.	1 дискета + документация	25	IBM PC
SL-04	Atoll	**Библиотека растровой графики для Pascal, C. Преобразование изображения в реальном времени, видеоэффекты, зумирование, анимация.**	1 дискета + документация	25	IBM PC EGA/VGA
SL-05	SPRight Lib	Объектно-ориентированная библиотека для Borland C++ 1.xx-2.xx. Спрайты, спрайтовые объекты, звуковые эффекты, поддержка графического формата PCX. Прозрачна для большинства графических библиотек. Поставка в исходных текстах.	1 дискета + документация	50	IBM PC/AT EGA/VGA
SL-07	**ПараФорт**	**Форт-система, поддерживающая стандарт FORTH-83. Турбосреда. Оконная библиотека. Встроенный форт-ассемблер.**	1 дискета + документация	30	IBM PC/AT
SL-08	Форт-Ягуар	32-разрядная форт-система для защищенного режима. Полное использование всей оперативной памяти компьютера. Встроенный форт-ассемблер, вещественная арифметика, турбо-среда, оконная библиотека.	1 дискета + документация	70	IBM PC 386
SL-09	Super Vision 3.0	**Интегрированный графический пакет для Borland Pascal 7.0, MS-DOS. Развитая иерархия объектов — элементов пользовательского интерфейса, интерактивный редактор ресурсов, средства для работы с растровыми графическими изображениями различных форматов.**	3 дискеты + документация	120	IBM PC/AT
SL-10	Графическая библиотека Парафорта	Работа с файлами PCX, спрайты, анимация, графические примитивы. Работа со шрифтами, включая создание и редактирование.	1 дискета + документация	30	IBM PC/AT
SC-02	TBG	**Библиотека для программирования графических приложений (BGI). Создание логической структуры изображения. Манипуляция изображениями сложной топологии. Графический редактор.**	1 дискета	20	IBM PC/AT EGA/VGA «мышь»
SC-03	Турбограф	Графический редактор с генерацией кода для компиляторов Borland. Визуальное программирование на Turbo Pascal, Turbo C/C++. Работа во всех графических режимах, поддерживаемых стандартной библиотекой.	1 дискета	20	IBM PC

Код	Продукт	Краткая характеристика	Объем поставки	Цена, $	Требованияк оборудованию
SU-01	FDA 6.1	Анализ и точное копирование гибких магнитных дисков всех типов. Самостоятельное создание ключевых меток любого уровня сложности. Быстрое и качественное форматирование всех видов дискет любого формата, в том числе нестандартного.	1 дискета	35	IBM PC/AT
SU-02	ALL.BGI	Набор русифицированных графических шрифтов в формате BGI. Набор драйверов в стандарте BGI: VGA 320×200×256, SVGA 640×400×256, 640×480×256, 800×600×256, 1024×768×256.	3 дискеты	10	IBM PC/AT
SU-03	FDREC	Программа восстановления поврежденных файлов на дискетах всех типов.	1 дискета	20	IBM PC AT
SU-04	Mouse Commander 1.03	Универсальный драйвер эмуляции «мыши» для систем, не обеспечивающих ее поддержку (APM, игры и так далее). Дополнительно набор утилит (русификатор и другие).	1 дискета + документация	5	IBM PC, PS/2/1, «мышь»
SU-05	Clipper & Clipper Tools 1 (Summer '87, рус.)	Популярный пакет разработки баз данных	пакет + документация	40	IBM PC/AT
MS-03	Слово и дело	Популярный отечественный текстовый редактор с богатыми функциональными возможностями	1 дискета	10	IBM PC, CGA
MS-04	Физика в картинках	Учебный компьютерный курс физики для школы. Включает три части: термодинамика и молекулярная физика, механика, электричество. Возможно приобретение отдельных частей.	2 дискеты + документация	75	IBM PC, EGA/VGA, «мышь» не обязательна
SG-01	Бизнес, Цена и спрос	Комплект учебных игровых программ по экономике, ориентированных на развитие навыков предпринимательства.	2 дискеты + документация	60	IBM AT-286, VGA, «мышь»
SG-02	Игры фирмы «ДОКА»	Набор развивающих увлекательных игр для DOS и Windows	10 наименований	10 за ед.	IBM PC/AT, EGA/VGA

Компьютерный ларек работает ежедневно, кроме субботы и воскресенья, с 11 до 17.
Адрес для писем: 117419 Москва, а/я 765.
Тел: (095) 238-99-33, 238-31-89.
Факс: (095) 238-99-11.

НОВОСТЬ!

ПРОКАТ ИГР НА CD-ROM

По всем вопросам обращаться
в редакцию
238-99-33

Компьютерные решения в гуманитарном образовании

А. Горчаков

На конкретном примере рассматриваются некоторые аспекты использования персональных компьютеров в гуманитарном образовании.

«Аппетит приходит во время еды» — гласит народная мудрость. По мере расширения возможностей персональных компьютеров, поначалу созданных для решения довольно простых задач, растут и потребности пользователя в применении ПК в предметных областях. Происходит неизбежный переход от решения отдельных, хотя и сложных, задач к использованию ПК для комплексной автоматизации деятельности. Одной из самых сложных, с точки зрения совокупности решаемых задач, предметных областей является образование. В настоящей статье будет рассказано об опыте организации компьютерного класса на философском факультете МГУ.

При принятии руководством факультета решения о приобретении такого класса, основными направлениями деятельности были признаны: формирование у студентов навыков работы с компьютером, внедрение ПК в учебный процесс, оказание помощи сотрудникам и студентам факультета в наборе и оформлении печатных материалов, поддержка издательской деятельности факультета, участие в международном информационном обмене. В соответствии с этими целями были определены требования, которым должны были удовлетворять приобретаемые компьютеры: соответствовать современному мировому уровню, использовать разнообразные учебные программы (хотя бы и на английском языке), использовать другие технические средства обучения (такие как видео и аудиоаппаратуру), иметь удобную и мощную среду для разработки собственных учебных программ, обеспечивать максимальное удобство при создании печатных материалов высокого качества, легко интегрироваться в мировую информационную сеть. Важными характеристиками были признаны также соответствие мировым стандартам по безопасности, минимальное время обучения навыкам работы с компьютером, наличие эффективной сервисной поддержки в процессе эксплуатации и возможной модернизации оборудования.

На основе этих требований было принято решение о приобретении компьютеров Macintosh фирмы «Apple Computer». Ряд особенностей этих компьютеров существенно упрощает процесс освоения последних. Так, в поставляемую вместе с ПК русифицированную операционную систему включена программа «Основы Макинтоша» («Macintosh Basic»), целью которой является обучение пользователя основным приемам работы с программами. ПК Macintosh удовлетворяют самому строгому (шведскому) стандарту безопасности, для них имеется большой выбор программ в различных областях знания. Компьютеры фирмы «Apple» общепризнаны в издательской деятельности, а AV-модели имеют все необходимое для использования аудио- и видеоаппаратуры и, благодаря GeoPort-архитектуре, подключаются в любую сеть.

Однако выбор платформы еще не означает выбора конкретной комплектации. На этом этапе очень важно правильно

выбрать будущего поставщика оборудования. В нашем случае в качестве партнера была выбрана фирма «Septem». Такой выбор был сделан на основании информации, собранной одновременно с определением требований к компьютерному классу. Эта фирма является субдистрибьютером (в Москве их всего два) компании «RUI Apple Computer IMC», представляющей интересы «Apple Computer» в России, имеет четырехлетний опыт продажи компьютеров Macintosh (то есть начала свою деятельность почти одновременно с первым появлением их в нашей стране и первой заключила дилерское соглашение с тогдашним представителем «Apple Computer»). Она поставляла эти ПК на такие известные отечественные предприятия, как ГАЗ, АЗЛК, шахта «Распадская» и другие, и, наконец, что крайне важно, имеет статус дилера в образовании и опыт работы с учебными учреждениями. Здесь уместно напомнить, что оборудование, поставляемое в сферу образования фирмой «Apple Computer» через своих дилеров, отпускается по специальным ценам, причем при такой поставке дилер не вправе извлекать из нее прибыль, а может лишь установить наценку, связанную с покрытием собственных расходов (это не относится к НДС, СН и таможенной пошлине, взимаемых нашим государством в полном объеме). Система скидок построена таким образом, чтобы привлечь внимание к более современным компьютерам, стоимость которых может составлять примерно 50% от розничной, если планируется использовать их в области образования. Цены же на более ранние модели почти не отличаются друг от друга.

Исходя из направлений деятельности компьютерного класса и имевшейся в распоряжении факультета суммы, совместно со специалистами фирмы «Septem» были определены несколько возможных комплектаций. Отметим тот факт, что фирмой «Septem» был предложен не просто набор определенной техники, а решение, удовлетворяющее задачам заказчика, что впоследствии полностью подтвердила практика. Важнейшим направлением считалось введение ПК в учебный процесс, поэтому все варианты базировались на объединении приобретаемых компьютеров в локальную сеть. Ознакомившись с поставленными задачами и предполагаемым размещением ПК, специалисты фирмы рекомендовали использовать для этого локальную сеть на основе EtherNet. В качестве центральной была предложена модель Macintosh Quadra 660 AV с диском 500 Мб, CD-ROM и 16 Мб RAM, в качестве локальных компьютеров — Macintosh LC475 (диск 160 Мб, 8 Мб RAM). В целях экономии средств заказчика, а также учитывая наличие на факультете машин типа IBM PC/AT, в качестве устройства печати фирмой был предложен сетевой принтер Hewlett-Packard 4 ML, который может работать с обеими платформами. Различия в вариантах комплектаций объяснялись возможными вариациями суммы, направленной на эти цели, и касались изменения некоторых характеристик приобретаемых ПК, но не затрагивали принципиальных решений. Для того чтобы учебный процесс мог проводиться максимально эффективно, в качестве сетевого программного обеспечения был выбран пакет Timbuktu компании «Farallon».

АППАРАТУРА

Компьютерный класс — дело не простое, особенно в образовании. Однако предоставляемые им возможности могут существенно изменить учебный процесс, сделать его более интересным.

Теперь о главном. Были ли достигнуты поставленные цели? Сегодня, спустя полгода после начала работы компьютерного класса, можно ответить на этот вопрос положительно. Благодаря программе «Основы Макинтоша», процесс обучения студентов начальным навыкам работы с компьютером удалось сделать весьма гибким, не требующим от преподавателей большого объема чисто технической, рутинной работы. Студенты приходят в класс в удобное для них время и самостоятельно осваивают основные приемы работы на ПК. В тех случаях, когда возникают какие-либо вопросы (что бывает крайне редко), они прибегают к помощи специалистов, отвечающих за работу класса. Аналогичным образом ведется и подготовка преподавателей-предметников, которым предстоит проводить занятия по своим дисциплинам с использованием учебных программ. Выбор таких программ преподавателями происходит на основании просмотра соответствующих каталогов, ознакомления с демонстрационными версиями программ, предоставленных факультету отделом образования компании «RUI Apple Computer IMC», и проведения по демонстрационным версиям пробных занятий.

Следует более подробно рассказать о методике проведения

занятий с использованием ПК. После апробации различных форм был выработан следующий методический подход. В начале занятия педагог сообщает тему и цели занятия. Далее на центральном компьютере он показывает учащимся тот раздел программы, с которым они сегодня будут работать, и демонстрирует правила работы с программой, после чего она запускается на локальных ПК, и студенты приступают к работе. Преподаватель имеет возможность полностью контролировать работу студентов, используя свой компьютер. Конечно, это лишь общая схема проведения семинаров, на каждом конкретном занятии в нее вносятся определенные коррективы.

Такая методика стала возможной благодаря объединению компьютеров в локальную сеть и использованию, по совету фирмы «Septem», программы «Timbuktu». Когда преподаватель объясняет группе тему занятия и приводит примеры, он делает это на центральной ПК. Локальные машины в это время подключаются к центральной в режиме просмотра, и на экране каждой из них происходит то же, что и на экране центрального компьютера. Когда студенты приступают к самостоятельной работе, уже центральный компьютер подключается к локальным в режиме просмотра или контроля (по желанию педагога), причем такое подключение может быть осуществлено одновременно ко всем ПК. Переходя из окна в окно, педагог может быстро проверить работу студентов, и, если подключение осуществлено в режиме контроля, помочь им, а при необходимости исправить их ошибки. Студенты имеют также возможность работать с учебными программами самостоятельно, посещая класс в свое свободное время. При отработке методики преподавания у сотрудников факультета не возникло каких-либо существенных трудностей, связанных с работой аппаратуры. В процессе работы обнаружилась возможность реализации таких идей, которые не могли возникнуть заранее. По мере ознакомления персонала с возможностями компьютеров, постоянно расширялся круг решаемых задач, используемых приемов, рабочих материалов. Не было ни одного случая, когда проявилась бы какая-нибудь некомплектность в подборе оборудования. Через два месяца после начала работы в классе был проведен экспериментальный открытый урок, на котором присутствовали представители различных российских ВУЗов.

Что касается издательского аспекта работы класса, упомянем лишь о том, что каждая из машин LC475 поставляется с русифицированным текстовым редактором WordWrite, и, следовательно, каждый пришедший в класс для работы над текстами имеет возможность, ознакомившись с документацией, создавать свои текстовые материалы, иллюстрируя их по мере необходимости. Для более сложных случаев предполагается использовать программу Microsoft Word, что привлечет людей, имеющих опыт подготовки текстов на IBM PC/AT. Важным моментом работы с текстами является возможность переносить их с платформы на платформу. Опыт показывает, что наличие на Macintosh соответствующих конверторов целиком решает эту проблему.

Важную роль в современном мире играет информационный обмен. Именно поэтому для более эффективной подготовки специалистов необходимо иметь доступ к самой свежей информации. Такой доступ проще всего получить, если воспользоваться возможностями международной электронной сети, например такой, как Internet. В этом случае участник информационного обмена имеет возможность оперировать информацией в соответствии со своими правами доступа, причем она может находиться в любой точке земного шара. В силу того, что информация поступает по спутниковым каналам связи, важно, чтобы линия между ПК пользователя и сервером работала максимально быстро. Обычно для этого используются так называемая выделенная линия и два (по одному на каждом конце линии) модема со скоростью передачи до 115 кбит/с. Поскольку зарубежные варианты таких модемов очень дороги, были приобретены их отечественные аналоги. Такое решение было принято после консультации со специалистами фирмы «Septem», взявшимися безвозмездно выполнить работы по их подключению к ПК Macintosh, что подтвердило высокий уровень сервисной поддержки фирмой своих клиентов.

«Не хвались на рать идучи, а хвались с рати едучи» — гласит еще одна народная мудрость. Теперь, когда пройден предварительный этап введения ПК в учебный процесс, можно сказать, что опыт факультета в использовании компьютеров в образовании может быть полезен другим, потому что представляет собой возможный пример использования новых форм в образовании, взаимодействия государственной и коммерческой организации (еще раз поблагодарим фирму «Septem») и отработки методического аспекта преподавания предметных дисциплин с использованием ПК. ■

КОНТАКТЫ

Телефон компьютерного класса философского факультета МГУ: (095)939-5701.

Фирма «Septem»
Телефоны: (095)231-3969, (095)233-4383, факс: (095)231-9241, e-mail: septem@applelink.apple.com

Новый стандарт печати — 1200 точек на дюйм

В феврале 1995 года компания «Lexmark» объявила о выпуске серии лазерных принтеров «Optra», использующих принципиально новую технологию лазерной печати, позволяющую добиться самого высокого аппаратного разрешения печати. Существенное отличие принтеров «Optra» — реальное разрешение 1200 на 1200 точек, а не его эмуляция. Это стало возможным только благодаря многим усовершенствованиям в технологиях, таким как бездисперсный тонер 8 микрон, новое устройство переноса тонера и новое печатающее устройство. Принтеры этой серии снабжены 25 МГц RISC-процессором AMD, оперативной памятью до 64 Мб, флэш-памятью для хранения шрифтов, макросов, графики до 4 Мб и жестким диском на 40 Мб. Все модели принтеров имеют два разъема для встроенных сетевых адаптеров «Lexmark MarkNet» и «MarkNet XL», поддерживающих одновременно до 18 операционных систем и выпускаемых в зависимости от типа сети в модификациях Ethernet, Token-Ring и LocalTalk. Принтеры «Optra» поставляются в комплекте с утилитой «MarkVision», которая дает возможность контроля и управления всеми принтерами в сети. Можно использовать бумагу плотностью от 60 до 300 г/кв.м. Стоимость новых принтеров держится на уровне 2300 долларов, что не отличается от цены хороших 600-точечных принтеров.

Версия 3.0 редактора PictureMan

В декабре 1994 года российской фирмой «STOIK Software» представлена новая версия графического редактора PictureMan 3.0. PictureMan представляет собой систему обработки изображений и видео под Windows на русском языке. Компанией STOIK объявлено также о выпуске средств дополнительной поддержки фильтров в стандарте Adobe PhotoShop Plug-In и модуля OCR фирмы «Cognitive Technologies».

Инструментарий MFC Migration Kit компаний «Microsoft» и «Symantec»

Компании «Microsoft» и «Symantec» совместно представили инструментарий MFC Migration Kit, облегчающий переход программистов к использованию технологии MFC. Инструментарий предлагает программистам, пишущим на C, набор утилит для переноса их C-кода в объектно-ориентированную среду разработки MFC, автоматически обеспечивая разработчикам совместимость 16 и 32-разрядных приложений. Компания «Symantec» планирует включить библиотеку классов MFC 3.0 в состав разрабатываемой сейчас новой объектно-ориентированной системы программирования на C++, а также представить дополнительные инструментальные средства для компиляторов Microsoft Visual C++™. Пакет Migration Kit предназначен для работы с компиляторами Visual C++, Symantec C++, Watcom C/C++, MetaWare High C/C++.

Новые Borland C++ 4.5 и DataBase Tools

Компания «Borland International» объявила о выпуске новой версии высокопроизводительного компилятора для DOS, Windows 3.1, Windows NT, Windows 95 и комплекта для создания приложений, работающих с базами данных Borland C++ 4.5 и Borland C++ 4.5 DataBase Tools. Компилятор обеспечивает разработчика эффективными средствами визуального создания приложений в ANSI-стандарте языков C и C++. Новые средства OLE Fast Track позволяют включать технологию OLE 2.0 в создаваемые приложения. Сохранены настраиваемая интегрированная среда разработки, библиотека объектов OWL 2.0, Turbo Debugger®, Turbo Profiler®, встроенная поддержка VBX. В комплект DataBase Tools, кроме компилятора, входят Borland Database Engine и Borland SQL Links. Они обеспечивают работу с данными во всех основных форматах, а также с базами данных через ODBC-драйверы. Компилятор предлагается по цене 350-400 долларов при стоимости upgrade 200-240 долларов, а пакет DataBase Tools — по цене 600-650 долларов при upgrade 350-450 долларов.

«Borland Simplify» обявляет о выпуске Dashboard 2.0 для Windows

«Borland Simplify», подразделение компании «Borland International», объявило в конце 1994 года о выпуске Borland Dashboard 2.0 для Windows. Пакет представляет собой расширенный набор утилит, помогающих пользователю полностью контролировать рабочую среду: осуществлять поиск программ и папок, быстро вызывать часто используемые приложения, следить за использованием системных ресурсов. Пакет распространяется по цене около 50 долларов.

Награда OS/2 Warp

14 ноября 1994 года еженедельник «Infoworld» присвоил OS/2 Warp новую награду «Hot Pick». Этой награды удостаиваются продукты, выдающиеся по техническим возможностям или лучшие в классе и прошедшие тестирование специальным коллективом. Среди основных особенностей продукта выделяются улучшенная поддержка технологии OLE, увеличение более чем на 15% скорости работы по сравнению с версией 2.11, полное соответствие Workplace Shell стандарту GUI. При уста-

новке операционной системы ею было обнаружено и правильно сконфигурировано более 75% существующих приложений. При тестировании отмечены многозадачные возможности новой версии операционной системы. На момент тестирования не существовало ни одной операционной системы, поставляющей так много приложений в своем составе, как OS/2 WARP. По шкале еженедельника продукту была выставлена оценка 8.1. 20 декабря 1994 года представители компании IBM сообщили, что за полтора месяца продано 800000 копий новой операционной системы.

Облегчение работы с сетями с помощью The Norton pcANYWHERE

Компания «Symantec» объявила в декабре 1994 года о выпуске новой версии пакета The Norton pcANYWHERE 2.0 для Windows, обеспечивающей более быстрый и простой доступ к удаленным компьютерам, а также локальным и корпоративным сетям. Помимо стандартной поддержки DOS и Windows™, версия 2.0 пакета полностью соответствует спецификациям новой операционной системы Microsoft Windows 95. Новая версия включает расширенную справочную систему, режим The Norton Assistant, консультирующий пользователя во время установки и конфигурации пакета, и «всплывающие» подсказки Cue Cards, детализирующие справочную информацию. Почти на 25% повышена эффективность удаленного доступа, значительно улучшен режим передачи данных. Версия 2.0 поддерживает протокол TCP/IP, а также позволяет работать в режиме двустороннего коммуникационного шлюза. Ориентировочная цена пакета The Norton pcANYWHERE – 150 долларов, upgrade с предыдущих версий пакета для DOS и Windows можно сделать за 70 долларов.

Централизованное управление с помощью Access Server 1.0

Компания «Symantec» представила новый пакет The Norton pcANYWHERE Access Server 1.0, предлагающий сетевым администраторам быстрый, экономичный способ централизованного управления несколькими сеансами удаленного доступа к компьютерам, а также локальным и корпоративным сетям. Пакет поддерживает до 8 одновременных сеансов удаленного доступа через модем или по сети. The Norton pcANYWHERE Access Server объединяет автономную и сетевую версию в одном пакете и предлагается по цене 330 долларов.

Enterprise Developer 1.1 от «Symantec»

Компания «Symantec» объявила о выпуске очередной версии инструментария разработки приложений Enterprise Developer, представляющего новое поколение инструментальных средств создания сложных распределенных приложений, работающих с корпоративными базами данных в рамках архитектуры «клиент-сервер». Используя архитектуру SCALE, разработчики смогут создавать приложения за меньший срок. Применение высокоэффективных объектов SCALE C++ резко сократит размер кода и время разработки. Пакет позволяет гораздо быстрее создавать сложные экранные формы. Крупные приложения «клиент-сервер» могут создаваться без программирования с использованием утилиты Rapid Forms Builder и визуальных инструментальных средств. Формы автоматически оснащаются полным набором встроенных функций, включающих ввод данных, поиск, просмотр, выборку, печать и выдачу запросов. Среди новых возможностей версии 1.1 поддержка интеграции электронной почты и создание локальных SCALE репозиториев с использованием базы данных Watcom. Версия Enterprise Developer для одного разработчика стоит 795 долларов, цена модификаций и пакета для нескольких пользователей колеблется от 1000 до 2500 долларов.

Модернизация средств персональной конференц-связи ProShare

В конце 1994 года корпорация «Intel» объявила о расширении семейства средств персональной концеренц-связи ProShare путем включения в него средства видеоконференц-связи через локальные сети в дополнение к цифровым сетям ISDN. «Intel» также внедрила приставку к своему семейству систем управления LANDesk Manager, которое позволяет администраторам сетей развертывать средства видеоконференц-связи через существующие корпоративные сети. Эти усовершенствования позволяют конечным пользователям осуществлять связь в пределах существующих инфраструктур, не нарушая режима их работы. Для развертывания систем телеконференц-связи на базе ПК в рамках корпоративной сети «Intel» предлагает устройство управления персональной конференц-связью LANDesk Personal Conferencing Manager. С помощью этого продукта администратор ЛАН может гарантировать выполнение существующих сетевых заявок, используя при этом преимущества телеконференц-связи. Все продукты системы ProShare обслуживают самые распространенные сетевые протоколы, включая IPX, TCP/IP, NetBIOS, и любой тип сети, а также широкие районные соединения T-1, T-3, Frame Relay. Видеосистема ProShare Video System 150 поступила в продажу по цене 1999 долларов США. ■

Все упомянутые в блоке новостей торговые марки, зарегистрированные торговые марки, товарные знаки и зарегистрированные товарные знаки принадлежат их уважаемым владельцам.

ВИМКОМ

Системный интегратор

КОРПОРАТИВНЫЕ СЕТИ
от А до Я

* Сетевое оборудование LAN 10-100 Мбит/с
  WAN 9.6-2048 Кбит/с
* Оптоволоконные технологии
* Серверы и Рабочие станции
  Hewlett-Packard 9000 и PC
* Сетевые операционные системы
  NOVELL NetWare, UNIXWARE, SCO/UNIX
* СУБД Informix, Oracle
* Электромонтажное оборудование
  Legrand

МОСКВА	(095)306 4101	Орел	(08600)25 317
С-Петербург	(812)316 3663	Брянск	(0832)55 2184
Ставрополь	(86522)41 855	Саратов	(8452)24 3117
Липецк	(0742)72 0200	Н-Новгород	(8312)22 6510

НОВЕЙШАЯ КОМПЬЮТЕРНАЯ ТЕХНИКА И ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ

СИСТЕМНАЯ ИНТЕГРАЦИЯ

- Комплексные решения в области создания корпоративных информационных систем
- Интеграция имеющихся информационных ресурсов в единую систему
- Разработка и моделирование сетевых проектов в нашей лаборатории
- Оперативная техническая и информационная поддержка
- Надежное гарантийное и послегарантийное обслуживание
- Обучение персонала и консультирование
- Предварительное апробирование решений

ЛОКАЛЬНЫЕ И КОРПОРАТИВНЫЕ КОМПЬЮТЕРНЫЕ СЕТИ

- Оборудование Ethernet 10/100, Token Ring, FDDI, ATM
- Многофункциональные модульные концентраторы
- Мосты и мультипротокольные маршрутизаторы
- Телекоммуникационное оборудование
- Диагностическое оборудование: кабельные сканеры, рефлектометры, протокольные анализаторы. Средства администрирования, мониторинга и управления на базе SNMP

ВЫСОКОПРОИЗВОДИТЕЛЬНЫЕ ОТКАЗОУСТОЙЧИВЫЕ СЕРВЕРЫ

- Серверы семейства ProLiant фирмы COMPAQ
- Модульные процессорные карты 486/Pentium с архитектурой SMP позволяют создавать наращиваемые 2-х и 4-х процессорные комплексы с симметричной обработкой
- Оперативная память до 512 МБ с корректирующими кодами
- Автономная система диагностики и контроля в процессе работы со встроенным SNMP-агентом
- Compaq TriFlex с пропускной способностью 267 МБ/сек.

МНОГОУРОВНЕВЫЕ СИСТЕМЫ ХРАНЕНИЯ ДАННЫХ

- Дисковые массивы RAID-5
- Магнитооптические накопители и роботизированные библиотеки
- Стриммеры QIC, DAT, DLT, 8mm Helical Scan
- Ленточные библиотеки CD-ROM: Readers, CD-Recorders, Changers, Premastering Software. Интегрированные системы архивации и миграции данных в среде NetWare, Unix, Windows NT

COMPAQ
Авторизованный системный партнер

CROC
INCORPORATED

Россия, 103050 Москва, ул. Ермоловой 22, стр. 1.
Тел.: (095) 200-1696, 299-4307
Факс: (095) 299-4625